

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



---

**Maestría en Ciencias  
en Electrónica y Telecomunicaciones  
con orientación en Telecomunicaciones**

---

**Enrutamiento de flujos en SDN utilizando optimización  
multiobjetivo**

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

Presenta:

**José Enrique González Trejo**

Ensenada, Baja California, México  
2019

Tesis defendida por  
**José Enrique González Trejo**

y aprobada por el siguiente Comité

---

**Dr. Raúl Rivera Rodríguez**  
Director de tesis

Miembros del comité

**Dr. José Eleno Lozano Rizk**

**Dr. Salvador Villarreal Reyes**

**Dr. Gabriel Alejandro Galaviz Mosqueda**

**Dr. Rodrigo Méndez Alonzo**



---

**Dr. Daniel Saucedo Carvajal**  
Coordinador del Posgrado en Electrónica y  
Telecomunicaciones

---

**Dra. Rufina Hernández Martínez**  
Directora de Estudios de Posgrado

*José Enrique González Trejo © 2019*

*Queda prohibida la reproducción parcial o total de esta obra sin el permiso formal y explícito del autor y director de la tesis.*

Resumen de la tesis que presenta **José Enrique González Trejo** como requisito parcial para la obtención del grado de Maestro en Ciencias en Electrónica y Telecomunicaciones con orientación en Telecomunicaciones.

### **Enrutamiento de flujos en SDN utilizando optimización multiobjetivo**

Resumen aprobado por:

---

Dr. Raúl Rivera Rodríguez  
**Director de tesis**

Las redes de telecomunicaciones en la actualidad se han vuelto indispensables tanto para el desarrollo como bienestar del ser humano. Estas son empleadas cotidianamente por los medios de comunicación y aplicaciones de entretenimiento, como también por aplicaciones industriales, médicas y educativas. La función principal de una red consiste en transmitir información de un punto a otro, para ello se emplean diversas rutas en la red y su desempeño se define por las diversas condiciones en las trayectorias como: la capacidad del caudal y los retardos entre nodos. Estos factores son considerados indispensables en el proceso de enrutamiento orientado a la calidad de servicio (QoS) y tiene como objetivo garantizar el rendimiento de las aplicaciones que requieren de un trato preferencial para su desempeño, pero esto comúnmente, no suele proporcionarse en las redes convencionales por su alta complejidad y la mala asignación de recursos en el proceso de enrutamiento, convirtiéndose en uno de los principales problemas y retos para las prestaciones de servicios y aplicaciones que requieren recursos adicionales para su operación adecuada. Una solución a estos problemas es la implementación de redes definidas por software (SDN) al tener la capacidad de monitorear los requerimientos de las aplicaciones y los recursos disponibles antes de realizar el proceso de asignación de rutas en la red. La aportación de este documento consiste en abstraer los parámetros de la red SDN y emplearlos en conjunto para proporcionar una ruta favorable con QoS empleando un algoritmo genético en el proceso de enrutamiento por su eficacia al resolver problemas multiobjetivo.

**Palabras clave:** enrutamiento, SDN, QoS, optimización multiobjetivo

Abstract of the thesis presented by Jose Enrique Gonzalez Trejo as a partial requirement to obtain the Master of Science degree in Electronic and Telecommunications with orientation in Telecommunications

### **Multiobjective optimization for flow routing in SDN**

Abstract approved by:

---

Dr. Raul Rivera Rodriguez  
Thesis Director

Telecommunications networks today have become indispensable use both for the development and human well-being. These are used daily by the media and entertainment applications, as well as industrial, medical and educational applications. One of the main function of a network is transfer data information a point-to-point; for this, it uses different paths and its performance is defined by the various conditions of the routes such as: the capacity of flow and delays between the nodes; These factors are considered indispensable in quality of service routing (QoS) and aims to guarantee the performance of applications that require preferential treatment for their performance, but this is usually not provided in conventional networks by its high complexity and poor allocation of resources in data flows, becoming one of the main problems and challenges for the provision of services and applications that require additional resources for proper operation. A solution to these problems is the implementation of software defined networking (SDN) by having the ability to monitor the requirements of the applications and the resources available before carrying out the process of assigning routes in the network. The contribution of this document consists in abstracting the parameters of the SDN network and using them together to provide a favorable route with QoS using a genetic algorithm for its efficiency solving multi-objective problems

**Keyword: Routing, SDN, QoS, multiobjective optimization**

## **Dedicatoria**

A mi familia por su apoyo incondicional y sus consejos.

## Agradecimientos

Al CICESE por otorgarme una oportunidad, una formación académica de calidad y brindarme las herramientas necesarias para mi crecimiento académico, profesional y personal.

Al CONACYT por brindarme el apoyo económico para realizar mis estudios de maestría con número de becario **634080**.

Quiero agradecer ampliamente a mi director de tesis, el Dr. Raúl Rivera Rodríguez y al Dr. José E. Lozano Rizk por su orientación y guía durante mi investigación. Por su confianza, consejos y la gran aportación que brindaron a mi trabajo de tesis.

A los miembros de mi comité de tesis: Dr. Gabriel Alejandro Galaviz Mosqueda, Dr. Salvador Villareal Reyes, Dr. Rodrigo Méndez Alonzo por sus consejos, comentarios y observaciones que ayudaron a mejorar y enriquecer este trabajo de investigación.

A mis padres y hermanos por sus apoyos. Les debo mis logros.

A todos mis amigos y compañeros, por los buenos momentos y ayuda en la maestría.

# Tabla de contenido

	Página
<b>Resumen en español.....</b>	<b>ii</b>
<b>Resumen en inglés.....</b>	<b>iii</b>
<b>Dedicatoria.....</b>	<b>iv</b>
<b>Agradecimientos .....</b>	<b>v</b>
<b>Lista de figuras .....</b>	<b>ix</b>
<b>Lista de tablas .....</b>	<b>xi</b>
<b>Capítulo 1 Introducción</b>	
1.1 Antecedentes.....	1
1.2 Planteamiento del problema .....	3
1.3 Justificación.....	4
1.4 Objetivo general .....	5
1.5 Objetivos particulares.....	5
1.6 Hipótesis .....	5
1.7 Contenido de la tesis .....	6
<b>Capítulo 2. Redes definidas por software SDN</b>	
2.1 Introducción.....	7
2.2 Historia de las redes definidas por software .....	8
2.3 Arquitectura de las redes definidas por software .....	9
2.3.1 Capas .....	10
2.3.2 Interfaces.....	11
2.4 Controlador .....	12
2.4.1 Controlador OpenDaylight .....	14

2.5 Tablas de flujo OpenFlow .....	16
2.6 Calidad de servicio .....	17
2.6.1 Definición de parámetros de QoS en la red .....	18
2.6.2. Tasa de transmisión.....	19
2.6.3. Retardo.....	20
<b>Capítulo 3. Optimización multiobjetivo y algoritmos genéticos</b>	
3.1 Introducción.....	22
3.2 Características de la optimización multiobjetivo.....	23
3.3 Algoritmos genéticos .....	25
3.3.1 Características de los algoritmos genéticos .....	26
3.3.2 Métodos de selección .....	29
3.3.3 Métodos de cruzamiento .....	30
3.3.4 Métodos de mutación .....	31
3.3.5 Criterios de terminación.....	32
3.4 Algoritmo genético Mocell .....	32
<b>Capítulo 4. Propuesta de enrutamiento con algoritmo genético (mocell)</b>	
4.1 Introducción.....	35
4.2 Implementación del algoritmo genético .....	35
4.2.1 Representación de codificación propuesta para enrutamiento .....	36
4.2.2 Representación de la función fitness .....	38
4.3 Emulación de la red en Mininet.....	39
4.3.1 Topología.....	39
4.3.2 Implementación del Controlador OpenDaylight.....	42
<b>Capítulo 5. Resultados y conclusiones</b>	
5.1 Introducción.....	45

5.2 Evaluación del algoritmo genético .....	45
5.3 Conclusiones .....	55
 <b>Capítulo 6. Resultados y conclusiones</b>	
6.1 Introducción.....	56
6.2 Análisis e implementación del algoritmo genético.....	56
6.3 Análisis del algoritmo de enrutamiento en SDN.....	57
6.4 Aportaciones.....	58
6.5 Trabajo futuro.....	58
 <b>Literatura citada .....</b>	<b>60</b>
<b>Anexos.....</b>	<b>63</b>

## Lista de figuras

Figura	página
1	Eventos importantes en la historia de las redes definidas por software. ....9
2	Arquitectura de SDN (Akhilesh S et al., 2014). ....11
3	Componentes básicos de controlador SDN. ....13
4	Clasificación de controladores SDN. ....14
5	Descripción general de OpenDaylight. ....15
6	Estructura de una entra de flujo. ....16
7	Máxima tasa de transmisión en sesión. ....20
8	Frentes óptimos de Pareto en el mismo espacio de soluciones. . ....24
9	Diagrama de algoritmo evolutivo simple. . ....27
10	Representación gráfica del ciclo evolutivo para un GA. ....28
11	Operador de cruce simple. ....30
12	Operador con dos cruces de corte. ....31
13	Operador de cruce uniforme. ....31
14	Vecindarios, a) cuatro, b) ocho, c) doce. Para los individuos $I_1=(x,y)$ e $I_2=(x-1,y-1)$ . ....33
15	Representación del algoritmo genético Mocell con un conjunto no dominado de cuatro individuos. ....33
16	Topología propuesta para el ejemplo de representación de codificación. ....37
17	Topología de red propuesta para el problema de enrutamiento entre extremos. . ....40
18	Procedimiento de enrutamiento propuesto. ....42
19	Diagrama a bloques del funcionamiento del algoritmo genético propuesto. . ....44
20	Rutas del controlador A. Las rutas azules son posibles soluciones favorables, mientras que las rojas son rutas que no cumplen con el nivel de seguridad requerido ( $s > 3.2$ ). ....46
21	Rutas del controlador A. Delay (ms) / nivel de seguridad. ....46
22	Rutas del controlador A. Bandwidth (Mb/s) / nivel de seguridad. ....47
23	Rutas del controlador B. Las rutas azules son posibles soluciones favorables, mientras que las rojas son rutas que no cumplen con el nivel de seguridad requerido ( $s > 3.2$ ). ....47
24	Rutas del controlador B. Delay (ms) / nivel de seguridad. ....48
25	Rutas del controlador B. Bandwidth (Mb/s) / nivel de seguridad. ....48

26	Rutas del controlador A. En verde se muestran las rutas pertenecientes a la primera generación, en azul las rutas finales y en rojo las rutas que no cumplen con el nivel de seguridad. ....	49
27	Rutas del controlador B. En verde se muestran las rutas pertenecientes a la primera generación, en azul las rutas finales y en rojo las rutas que no cumplen con el nivel de seguridad. ....	49
28	Rutas: controlador, algoritmo genético, mayor ancho de banda y menor retardo.....	51
29	Tiempo de estimación de cada ruta para transferir 1 GB de información. . ....	51
30	Jitter en cada una de las rutas. ....	52
31	Datos promedio transferidos durante 200 segundos con los protocolos TCP y UDP. ....	53
32	Tasa de transmisión en cada una de las rutas durante 200 segundos. ....	54
33	Transferencia de datos TCP durante 1000 segundos.....	54

## Lista de tablas

Tabla	página
1	Tabla de adyacencias del controlador A para la representación de codificación. ....37
2	Métricas de los enlaces correspondientes al controlador A (Delay (ms), Nivel de seguridad, Jitter (ms) y ancho de banda (mbps)). ....41
3	Métricas de los enlaces correspondientes al controlador B (Delay (ms), Nivel de seguridad, Jitter (ms) y ancho de banda (mbps)). ....41
4	Rutas factibles de los controladores. ....50
5	Abstracción de nodos en mininet. ....63
6	Métricas de las rutas encontradas por el controlador A. ....66
7	Métricas de las rutas encontradas por el controlador B. ....67

# Capítulo 1 Introducción

---

## 1.1 Antecedentes

Hoy en día las redes de telecomunicaciones han llegado a ser imprescindibles en la vida cotidiana del ser humano, tanto para su bienestar como desarrollo social. Dichas redes brindan servicios para facilitar la comunicación y desarrollar aplicaciones para diversas áreas como la industrial, el educativa, la médica, hasta sectores comerciales y financieros por mencionar solo algunos.

Las redes de telecomunicaciones son un conjunto de dispositivos que conforman una infraestructura para el intercambio de información de un punto a otro; emplean un medio físico para transmitir información como cables o señales de radio que interconectan diversos dispositivos para conforman una red. Esta infraestructura requiere emplear protocolos de comunicación y dispositivos intermediarios para determinar las trayectorias adecuadas para enviar correctamente la información a su destino. Los protocolos de comunicación son los encargados de ejecutar de manera sistemática y distribuida las instrucciones para la toma de decisiones como es el caso de los algoritmos de enrutamiento. El proceso de enrutamiento inicia cuando un enrutador recibe una petición para transmitir un flujo de información y termina al establecer una trayectoria hacia su destino. Para esto, la información es fragmentada en pequeños bloques de datos que simplifican su distribución. Los paquetes de datos contienen la dirección de destino e información propia del protocolo de comunicación empleados para determinar una ruta y llegar a su destino.

En las redes tradicionales de telecomunicaciones como internet, se suelen emplear una gran diversidad de protocolos y algoritmos de enrutamiento para transmitir información. Sin embargo, la mayoría de ellos emplean esquemas de entrega de paquetes bajo el modelo del mejor esfuerzo, los cuales transportan de manera concurrente diversos tipos de flujo, como voz, video, datos y aplicaciones interactivas (croll, 2000). La entrega correcta de paquetes no garantiza que los flujos de información cuenten con las condiciones necesarias para brindar un servicio adecuado y no presenten problemas en su proceso de enrutamiento.

Los problemas más comunes son la pérdida de datos, retardos e interrupciones en algún punto de la ruta, sin mencionar los problemas de confiabilidad, como la alteración y modificación de datos, desvío, duplicación y robo de información (Bays et al., 2015), (Pattaranantakul et al., 2016) . Debido a esto, es

necesario que la infraestructura de la red soporte diversos mecanismos de enrutamiento robustos que además proporcionen confiabilidad en las rutas seleccionadas (Mao et al., 2017).

Para solucionar estos tipos de problema relacionados con el proceso de selección de rutas. Se emplean diversas métricas orientadas a la calidad de servicio (del inglés Quality of Service QoS). Durante más de diez años, muchos investigadores han estudiado la calidad de servicio en las redes y han desarrollado diferentes tipos de arquitecturas, tecnologías y mecanismos para su monitoreo y gestión (Firoiu et al., 2002), (Karam & Jensen, 2010) Internet Engineering Task Force (IETF por sus siglas en inglés) ha explorado y propuesto diferentes opciones para obtener la calidad de servicio requerida por las aplicaciones (Egilmez et al., 2012), por ejemplo, IETF RFC 1633 (Braden et al., 1994) , IETF RFC 2430 (Li & Rekhter, 1998) , IETF RFC 2475 (Blake et al., 1998), etc. Pero estas tecnologías no consideran medidas de seguridad para la protección de la información en su distribución por la red (Lamali et al., 2016). Para mitigar este riesgo, se han desarrollado diversos mecanismos de seguridad como: antimalware, firewalls y sistemas de detección de intrusos. Estas herramientas son empleadas como puntos de control y seguridad que se encuentran distribuidos estratégicamente en la red, sin embargo, con la escalabilidad de las redes, la diversidad de protocolos, aplicaciones y servicios dificulta su implementación y adaptación en los centros de datos y en las nubes computacionales.

Las redes definidas por software (SDN, del inglés Software Defined Network) surgen como una alternativa para facilitar la innovación de protocolos y superar las limitaciones de gestión con su control centralizado de la red y la optimización de rutas programables (Ramos et al., 2015). Convirtiéndose en un corto período de tiempo en una nueva tecnología alternativa para el futuro de las redes de telecomunicaciones al brindar un entorno dinámico, abierto y controlable (Henneke et al., 2016).

SDN es un nuevo paradigma de red que separa y abstrae el plano de control del plano de datos de los dispositivos de reenvío (routers y switches), proporcionando nuevas posibilidades para el desarrollo de aplicaciones, virtualización de redes, exploración de nuevos métodos de enrutamiento seguro, ingeniería de tráfico, gestión de recursos y dispositivos que mejoran la confiabilidad y la capacidad de administración y de la red (Nguyen & Yamada, 2016). El plano de control mantiene una vista global de toda la red y se encarga de tomar las decisiones, mientras que el plano de datos se utiliza solo para la transmisión de datos. Una de las tecnologías claves en SDN es OpenFlow (Jarraya et al., 2014), que define la interacción entre el controlador centralizado y el conmutador de datos para el funcionamiento del esquema de red centralizada.

Esta tecnología se ha convertido en una técnica prometedora para mejorar la eficiencia del enrutamiento en una red, el control de flujos de datos, esquemas de QoS y seguridad. Sin embargo, la investigación sobre las SDN está todavía en una etapa temprana y hay cuestiones que deben ser investigadas más a fondo. Por ejemplo, la búsqueda de rutas óptimas para aplicaciones que requieran calidad de servicio y seguridad, que representan problemas importantes que no han sido considerado ni estudiado plenamente por su compleja implementación (Basit & Ahmed, 2017).

## 1.2 Planteamiento del problema

En los últimos años el crecimiento en las redes de telecomunicaciones ha sido impredecible, particularmente en la nube por el incremento exponencial de dispositivos, aplicaciones, servicios y multimedia (contenido digital) que se agrega día a día. Se prevé que internet interconectará 50 mil millones de dispositivos para el 2020 (Djouani et al., 2019). Esto ha requerido el desarrollo y la implementación de diversos protocolos de enrutamiento para tratar de mitigar la demanda de recursos, proporcionar diferentes niveles de seguridad y calidad de servicio (QoS) para cada una de las aplicaciones manteniendo la integridad de la información. Sin embargo, proporcionar QoS se ha convertido en problema complicado por sus múltiples métricas, sin mencionar la gestión de los recursos de la red e implementación de seguridad en tales circunstancias en las redes convencionales.

El enrutamiento basado en QoS es complejo, dado que considera diversos parámetros claves para satisfacer las necesidades de los servicios y aplicaciones, como: ancho de banda, retardos, pérdidas de datos, consumo de energía, entre otras. Por lo cual, las redes modernas de telecomunicaciones requieren eficiencia en la elección de una ruta que proporcione la adecuada calidad de servicio (QoS), independientemente del tipo de flujo generado por una aplicación o servicio. Pero el enrutamiento de QoS, representa grandes desafíos debido a la complejidad de la optimización de múltiples objetivos en la ruta, como estados de red y conexión. Dichas condiciones no garantizan el cumplimiento de todos los requisitos determinados por las limitaciones de la red, la tasa de transmisión, los retardos en los dispositivos, la sobrecarga de tráfico en las rutas, la congestión en los nodos, la consideración de rutas seguras e integridad de los mecanismos de seguridad (firewalls, IDS, etc), al momento de realizar el enrutamiento (Papadimitratos & Haas, 2002).

Todos los requerimientos mencionados, exigen una gestión de nivel superior de la red e implementación de algoritmos más inteligentes para la optimización de rutas programables, centradas en la generación de rutas confiables que brinden calidad de servicio necesaria para las aplicaciones y servicios solicitados; por lo cual es necesario adoptar nuevos enfoques de administración orientado al software (Abe et al., 2015), (K. Li et al., 2018), (Ahmed & Ramalakshmi, 2018) como las redes definidas por software (SDN). Su plano de control es ideal para la planificación de rutas dinámicas orientadas a la calidad de servicio, por su entorno de red abierto, dinámico, y controlable.

Esta estrategia tiene casi el mismo propósito que los algoritmos de enrutamiento tradicionales, encontrar una ruta de acceso entre los diversos enrutadores y llegar al destino de la forma más rápida, pero al surgir múltiples flujos pueden aparecer problema de congestión y pérdidas de eficiencia en las aplicaciones al no considerar las condiciones de las trayectorias al momento de seleccionar las rutas, siendo el problema más común de los algoritmos de enrutamiento tradicionales como Dijkstra (Zhang y Yan, 2015). Permiten obtener una ruta corta mediante el conteo de saltos, sin embargo, no garantiza su eficiencia. Por lo cual es necesario emplear enrutamientos que consideren múltiples métricas siendo un aspecto importante de investigación (Doshi et al., 2018).

### **1.3 Justificación**

El enrutamiento se ha convertido en una tecnología clave para las redes de telecomunicaciones. Sin embargo, se centran en dos acciones básicas: determinar una ruta y transmitir datos a través de la red. Pero estos dejan de lado la consideración de los diferentes factores que determinan el comportamiento de las trayectorias sin mencionar su confiabilidad. Ningún sistema de enrutamiento tradicional integra la seguridad como un factor al momento de realizar el trazado de las rutas, convirtiéndose en un problema realmente alarmante y por consiguiente en un riesgo importante (Du, 2017).

Debido a que una red puede presentar diferentes dificultades por el manejo inadecuado del tráfico es necesario implementar nuevos algoritmos de enrutamiento que tengan presente el estado constante de la red y determinen rutas adecuadas y confiables para cada flujo, como son las redes SDN, sin embargo, al considerar diversas variables en el proceso de enrutamiento, aumenta su complejidad y dificultar

seleccionar una ruta óptima. Por ello, es necesario implementar métodos de optimización multiobjetivo adaptables con proceso de enrutamiento.

## **1.4 Objetivo general**

Proponer y diseñar un algoritmo de enrutamiento adaptativo con múltiples condiciones de QoS, empleando la tecnología de las redes definidas por software (SDN).

## **1.5 Objetivos particulares**

1. Investigar las principales técnicas de enrutamiento multivariable. Con la finalidad de identificar antecedentes, el estado actual y tendencias.
2. Investigar parámetros de calidad de servicio en redes de comunicaciones.
3. Emplear un algoritmo genético multiobjetivo y adaptarlo al proceso de enrutamiento.
4. Desarrolla la emulación de una red SDN para la evaluación del algoritmo de enrutamiento propuesto.

## **1.6 Hipótesis**

Empleando la tecnología de las redes definidas por software (SDN), es posible diseñar algoritmos de enrutamiento más inteligentes y robustos que además de incorporar la calidad de servicio para cada flujo de datos determinen rutas confiables para cada aplicación. La solución particular, es utilizar diversos puntos de seguridad en las redes como puntos de prioridad al momento de realizar el trazado de rutas y obtener rutas más confiables (Park et al., 2016), (Mao et al., 2017). Sin embargo, este método requiere de complicadas normas de direccionamiento para cada flujo de información. Siendo más sencillo la implementación de una nueva métrica relacionada con la confiabilidad de los nodos.

La consideración de utilizar una métrica adicional para la confiabilidad de cada nodo resulta ser una alternativa práctica y novedosa dado que esta no suele integrarse en el proceso de enrutamiento (X. Li et al., 2016). Al utilizar múltiples métricas en el proceso de enrutamiento los métodos tradicionales requieren de una evaluación exhaustiva y prolongada para encontrar una ruta favorable, tomando como alternativa la implementación de un algoritmo génico para el enrutamiento orientado a la calidad de servicio y seguridad por su eficiencia al resolver problemas con múltiples factores o variables en tiempos relativamente cortos como es el caso de este trabajo.

## **1.7 Contenido de la tesis**

Este documento se encuentra organizado de la siguiente manera: en el capítulo 2 se hace una descripción detallada de los conceptos de las redes definidas por software, cualidades de la calidad de servicio e implementación de tablas de flujo; en el capítulo 3 se describen los algoritmos multiobjetivo y su evolución a algoritmos genéticos, explicando sus características particulares y la descripción del algoritmo implementado en este trabajo; en el capítulo 4 se introduce la metodología para la solución del problema propuesto empleando un algoritmo génico para el proceso de enrutamiento orientado a calidad de servicio; en el capítulo 5 se presentan los resultados obtenidos del algoritmo y la emulación propuesta; y en el capítulo 6 se concluye mencionando el análisis del algoritmo, así como la aportación de este trabajo y las consideraciones para trabajo futuro de la presente investigación.

## Capítulo 2. Redes definidas por software SDN

---

### 2.1 Introducción

El escenario de las redes en la actualidad está creciendo rápidamente. Día a día las redes se expanden y se están volviendo cada vez más complicadas. Internet está consolidando la unión de las diversas redes de telecomunicaciones: datos, multimedia, telefonía, dispositivos móviles, sensores y demás, en una gran infraestructura de conectividad mundial, pero al mismo tiempo adquiere más desafíos principalmente en el enrutamiento, la integridad y la seguridad de la información. Siendo un reto proporcionar calidad de servicio QoS a las aplicaciones y aún más complicado satisfacer la demanda constante de los usuarios para adquirir una mejor calidad de experiencia (QoE por sus siglas en inglés).

La infraestructura de red tradicional se considera en gran medida altamente rígida y estática, ya que inicialmente se concibió para un tipo particular de tráfico de datos (texto), haciéndolo poco adecuado para los flujos multimedia interactivos y dinámicos sin mencionar los mecanismos de seguridad que no se establecieron. La capacidad de procesar y enviar datos en las redes de telecomunicaciones está siendo superada por el tráfico de datos, la demanda de servicios y la aparición de BigData (virtualización de servidores, contenidos multimedia, etc.), junto con el incremento en la utilización de dispositivos móviles y el internet de las cosas, han agregado complejidad a la arquitectura convencional de las redes, haciendo la administración de la información una tarea cada vez más difícil y especializada.

Las redes definidas por software (SDN) (Kreutz et al., 2015), nacen como un nuevo paradigma de arquitectura y enfoque para abordar los desafíos actuales y superar los problemas asociados con las redes convencionales. En particular las SDN simplifican las operaciones para gestionar y controlar la red, adaptándose a las condiciones de la red que cambian constantemente con el potencial de facilitar la programación y automatización de la configuración de la red al separar el plano de datos y de control de los dispositivos de red para mantener toda la lógica de control de los dispositivos centralizada para proporcionar un marco de configuración flexible para el desarrollo de aplicaciones especializadas y despliegue de nuevos servicios. En este capítulo se abordan las principales características de SDN y calidad de servicio.

## 2.2 Historia de las redes definidas por software

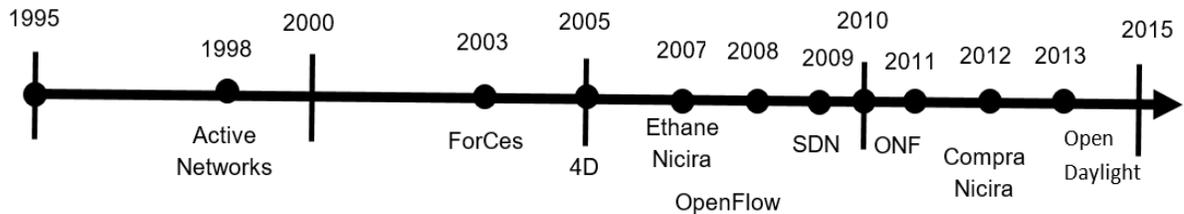
La red definida por software es un nuevo paradigma que permite separar la integración del plano de control y datos. El concepto SDN se introdujo por primera vez en junio del 2009 en la Universidad de Stanford, E.U.A. Siendo el resultado del trabajo consolidado por diferentes tecnologías ForCes (Doria et al., 2010), 4D (Greenberg et al., 2005), Ethane (Casado et al., 2007) con un fin en común, separar el plano de control del plano de transmisión de datos en los dispositivos de red y el objetivo de generar y evaluar nuevos protocolos de red más eficientes y rápidos de implementar.

El surgimiento, la consolidación de la separación de planos y la idea de mantener un punto de control como lo establece SDN, se muestra en la línea del tiempo de la Figura 1. Iniciando a partir de 1998 con la colaboración de Active networks al brindarles la capacidad de modificar y realizar cálculos a elementos de la red, siendo el primer intento en el desarrollo de dispositivos programables. Sin embargo, no establecieron un cambio significativo en la separación de planos. ForCes (Forwarding and Control Element Separation) es un trabajo presentado por el grupo de trabajo de la IETF (internet Engineering Task Force), destacando por centralizar la lógica de control al separar este plano con el de reenvío, pero se trataba de una centralización parcial, cada elemento de control interactuaba con su correspondiente elemento de datos. Este inconveniente, fue solucionado un año después por 4D al introducir un controlador centralizado para conectar todos los dispositivos y mantener de esta forma un sistema independiente y aislado dedicado únicamente para la gestión de la topología. Ethane y OpenFlow fueron otras tecnologías precedentes antes de que surgiera el término red definida por software.

Ethane es una solución basada en políticas que permiten acceso y administración de la red al otorgarle niveles de privilegios a sus usuarios, basándose en tres principios: (1) Las políticas rigen a la red ante cualquier decisión de los dispositivos. (2) El ruteo debe considerar las políticas establecidas y (3) la red debe conocer el origen de cada paquete (Casado et al., 2007). Siendo estas políticas precedentes a las reglas de flujo que permite establecer SDN.

OpenFlow (McKeown et al., 2008) es el estándar de facto de SDN, fue propuesto por el equipo de investigación Cleanslate de la Universidad de Stanford, definiendo un protocolo que permite a un controlador centralizado actuar como el plano de control, administrando el comportamiento de los dispositivos de la red de una manera dinámica. El controlador se comunica con los dispositivos de la red a través de una conexión segura, creando y administrando reglas de flujos de datos a través de una tabla.

Estas reglas se encargan de instruir a los dispositivos de la red sobre el manejo del tráfico y enrutamiento de flujos. Separando de esta manera el plano de control de los dispositivos. Esto dio origen a Open Networking Foundation (ONF), una organización dirigida por las principales empresas en el área de redes de computadoras como Google, Facebook, Yahoo!, Microsoft, Verizon y Deutsche Telekom, con el objetivo principal de promocionar y adoptar SDN, además de desarrollo desarrollo de estándares abiertos relacionados con esta tecnología (Agarwal et al., 2014).



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

La aceptación generalizada de SDN por parte de la academia y la industria han hecho que en poco tiempo este estándar sea exitoso y novedoso en la implementación de diversos entornos de redes, como redes comerciales y empresariales de gran escala y demanda, como la red de centros de datos de Microsoft (Hong et al., 2013) y la red troncal de Google por sólo mencionar dos de las más importantes a nivel global en la actualidad. Teniendo como resultado el interés colectivo para la estandarización de su arquitectura, capas y especificaciones de terminología para SDN en el RFC 74260 por parte de la Fuerza de Tareas de Investigación de Internet (IRTF, por sus siglas en inglés), (Haleplidis et al., 2015).

### 2.3 Arquitectura de las redes definidas por software

La red definida por software (SDN) surgió como una posibilidad de programar la red, en la cual el plano lógico de control (software) se desacopla del plano de reenvío de información (hardware), permitiendo de esta manera controlar, modificar y manejar el comportamiento de la red dinámicamente mediante software centralizado. A diferencia de las redes convencionales que requieren de la configuración individual de cada dispositivo, empleando comandos especializados para cada equipo y proveedor de servicio de la red que puede convertirse en un obstáculo para la escalabilidad y manejo de las redes.

Al implementar la arquitectura SDN, los dispositivos se vuelven más simples, agilizando el flujo de información para grandes volúmenes de datos y la resolución de problemas en la red. El controlador

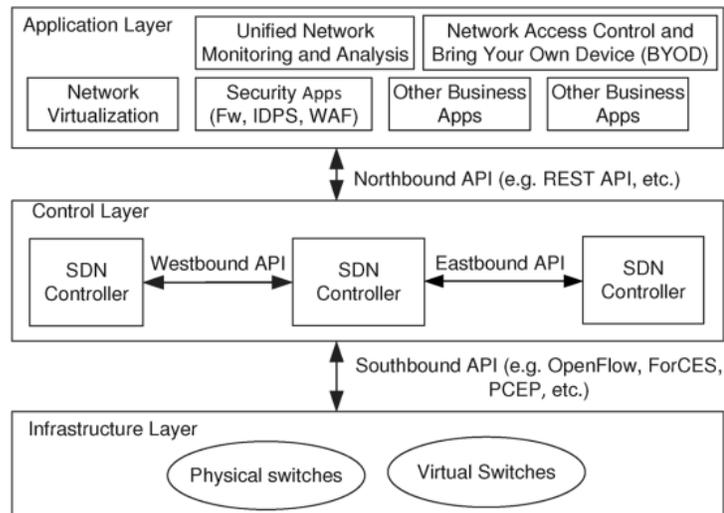
denominado cerebro de la red implementa la mayoría de las tareas de administración, servicios y selecciona de la ruta a través de la configuración de reglas de flujo dentro de los dispositivos de red, además de realizar un seguimiento de la información del paquete.

### 2.3.1 Capas

La OFN presenta en (Akyildiz et al., 2014) una arquitectura de alto nivel para SDN con tres capas principales:

- Capa de infraestructura: también conocida como el plano de datos. Consiste principalmente en elementos de reenvío, que incluyen conmutadores físicos y virtuales accesibles a través de una interfaz abierta que permite el cambio y reenvío de paquetes.
- Capa de control: también conocida como el plano de control, consiste en un conjunto de controladores SDN basados en software que proporcionan una funcionalidad de control consolidado de una interfaz abierta, teniendo tres interfaces de comunicación, las cuales permiten a los controladores interactuar entre las interfaces en dirección norte, sur y este/oeste. Estas interfaces se presentarán brevemente a continuación.
- Capa de aplicación: consiste principalmente en las aplicaciones del usuario final que utilizan las aplicaciones de SDN y los servicios de red, como la seguridad de la red, calidad de servicio, la ingeniería de tráfico, la gestión del control de acceso y el equilibrio de carga como servicio, entre otros servicios de virtualización de red.

La Figura 2, ilustra la arquitectura de SDN, mientras detalla algunas partes claves, como la capa de infraestructura, control y aplicación, así como las interfaces de comunicaciones entre estas capas.



**Figura 2.** Arquitectura de SDN (Akyildiz et al., 2014).

### 2.3.2 Interfaces

El controlador cuenta con tres interfaces, Northbound o hacia el norte (NBI, por sus siglas en inglés) que interactúa con el plano de gestión. Otra interactúa con el plano de datos llamada Southbound o hacia el sur (SBI, por sus siglas en inglés) y por último se encuentra la interfaz Eastbound/Westbound o hacia este/oeste (EBI/WBI, por sus siglas en inglés) que permite la comunicación con otros controlares en la red (Paliwal et al., 2018).

- Interfaz en dirección sur:** proporciona un medio de comunicación entre el controlador y los dispositivos de comunicación. Instala las reglas de flujo apropiadas en la tabla de reenvío de los dispositivos. OpenFlow (Narisetty et al., 2013) es el estándar de interfaz sur más ampliamente implementado de la comunidad de código abierto. Permite generar mensajes de eventos en casos de cambio de puerto o de enlaces al igual que permite realizar estadísticas basadas en el flujo de los dispositivos. En caso de que un paquete no cumpla con una regla de flujo, es enviado al controlador para determinar una acción. OpenFlow no es la única opción de interfaz hacia el sur, existen otras interfaces como: Open vSwitch Database (OVSDDB) (Pfaff & Davie, 2013), ForCes (Doria et al., 2010), OpFlex (Smith, 2014), entre otras.
- Interfaz en dirección Norte:** esta interfaz proporciona conectividad entre el controlador y las aplicaciones de red que se ejecutan en el plano de administración. A diferencia de la interfaz sur, está aún no cuenta con una estandarización común. Es una tarea crítica ya que el requisito para cada aplicación de red puede variar. Por ejemplo, una aplicación de seguridad puede tener

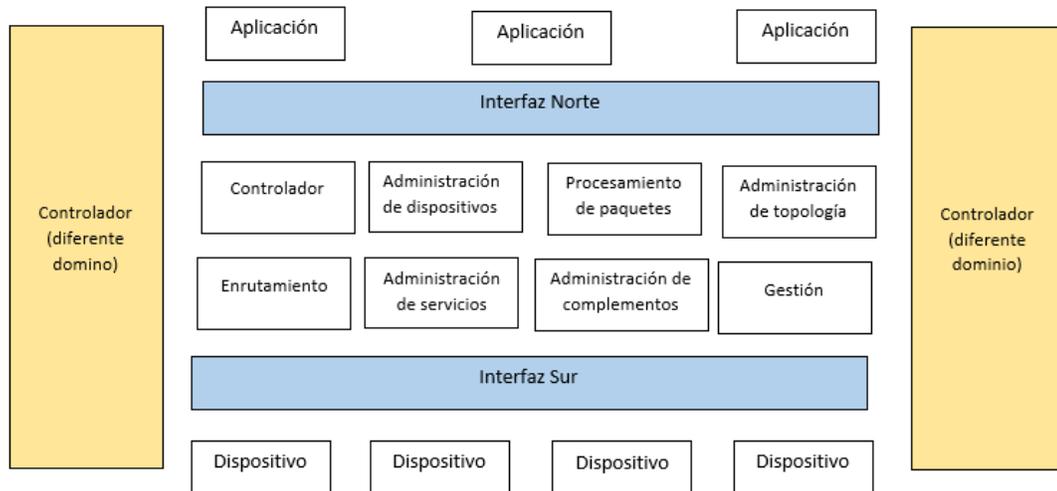
requisitos diferentes a una aplicación de enrutamiento. La ONF se encuentra trabajando en la estandarización, pero con el avance de la tecnología, se cuenta con una amplia gama de compatibilidad con API norte, como API ad-hoc y REST, siendo las más empleadas.

- **Interfaz en dirección este/oeste:** esta interfaz está destinada para la comunicación entre grupos de federaciones de controladores para sincronizar el estado de disponibilidad de los dispositivos.

## 2.4 Controlador

La inteligencia o pieza angular de SDN es el controlador, este se encarga de abstraer el estado de la red, permitiéndole de esta manera manipular y administrar los recursos y elementos de la infraestructura implícita. Traduce las peticiones de las aplicaciones mediante una API, construyendo tablas de flujo que posteriormente serán utilizadas por el plano de datos. La Figura 3, muestra los componentes principales de un controlador SDN, incluyendo sus módulos funcionales centrales, interfaces y un conjunto de aplicaciones comunes (Hoang y Pham, 2015):

- Unidad de gestión de controladores e interruptores: todos los dispositivos de red se encuentran enlazados con el controlador y puede determinar el estado de estos.
- Unidad de procesamiento de paquetes: procesa los encabezados de los paquetes de acuerdo con el protocolo de comunicación implementado; todos incluyen dirección de origen, destino, tipo de mensaje y carga útil.
- Administración de dispositivos: gestiona los dispositivos de la red; cada dispositivo esta modelado con una dirección de enlace de datos, red, conmutador y puerto.
- Administración de topología: se encarga de identificar cambios en la topología, manteniendo actualizado el estado de la red, además de enviar el estado de la topología a las aplicaciones.
- Enrutamiento: implementa las rutas de destino a partir de un protocolo.
- Capa de abstracción de servicios y complementos: es empleada para agregar nuevos protocolos al controlador.
- Interfaz de gestión: proporciona por lo general una aplicación web para acceder al controlador y permitirle al administrador acceder a las funciones del controlador de forma rápida y gráfica.



**Figura 3.** Componentes básicos de controlador SDN.

Las redes definidas por software utilizan dos tipos de controladores, los cuales son: controlador centralizado y controlador distribuido. La Figura 4, describe la clasificación de los distintos controladores en dos categorías centralizados y descentralizados (Bannour et al., 2018):

- Controlador centralizado: implementa toda la lógica del plano de control en una sola ubicación. Su principal ventaja es su simplicidad y administración, ya que proporciona un único punto de control. Sin embargo, sufren problemas de estabilidad porque cada controlador tiene una capacidad limitada para controlar dispositivos.
- Control distribuido: aparecieron como una solución potencial para mitigar los problemas provocado por las arquitecturas centralizadas (escalabilidad deficiente, punto único de falla y cuellos de botella). Como resultado se han propuesto dos modelos de arquitectura distribuida: una plana y otra de arquitectura jerárquica. La primera consiste en dividir la red en múltiples áreas, siendo controladas únicamente por un controlador. Este tipo de diseño permite una latencia de control reducida y una capacidad de recuperación ante problemas. La segunda arquitectura, divide la red en múltiples niveles dependiendo de los servicios requeridos, mejorando la escalabilidad de la red y rendimiento de la red.

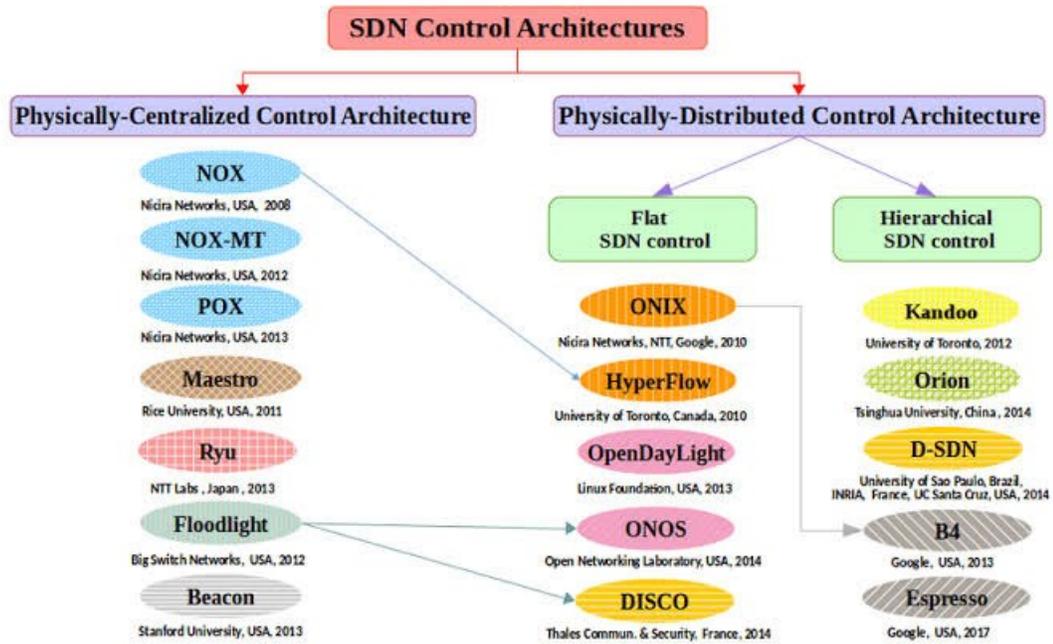


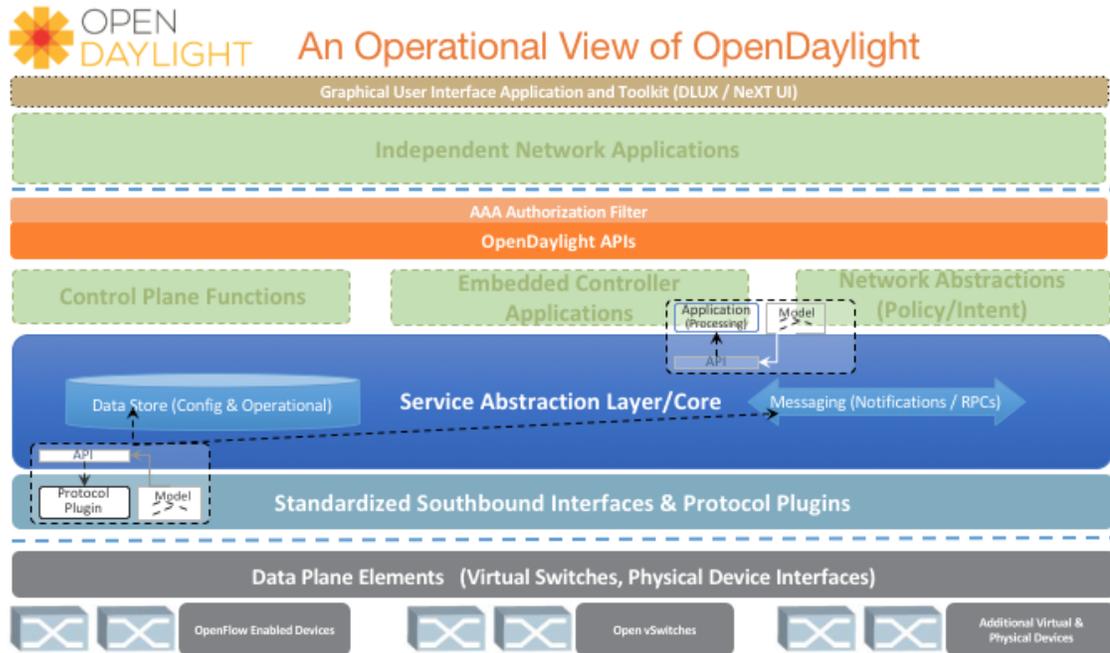
Figura 4. Clasificación de controladores SDN (Bannour et al., 2018).

#### 2.4.1 Controlador OpenDaylight

OpenDayLight (ODL), es una plataforma de controlador SDN de código abierto (Open Source) que cuenta con la capacidad de programar una red, permitiendo personalizar y automatizar redes de cualquier tamaño. Este controlador está implementado completamente en software y se encuentra contenido en su propia máquina virtual Java. Por esta razón, puede ser implementada en cualquier sistema operativo que soporte Java (OpenDaylight, 2019).

Como se muestra en la figura 5, la plataforma de OpenDaylight tiene la estructura de un entorno SDN, como lo hemos descrito en el subcapítulo 2.3. El controlador proporciona APIs (Interfaz de programación de aplicaciones por sus siglas en inglés) abierta hacia la interfaz norte que son utilizadas por las aplicaciones como OSGi (Servicio interactivo de Gateway abierto por sus siglas en inglés) y REST (Protocolo de configuración de red, por sus siglas en inglés). OSGi es un sistema que proporciona servicios directamente de la red al controlador, mientras que REST permite proporcionar información a aplicaciones externas y brindarles la capacidad de cambiar las reglas de la red.

En ODL, existen módulos dinámicamente conectados, responsables de realizar tareas en la red, pero también es posible insertar nuevos servicios y extensiones para mejorar la funcionalidad de SDN, como OpenFlow, OVSDb, NETCONF, BGP, por mencionar las más populares.



**Figura 5.** Descripción general de OpenDaylight (OpenDaylight, 2019).

Los dispositivos subyacentes y las aplicaciones de red están representados como objetos o modelos, cuyas interacciones se procesan dentro del servicio SAL (Servicio abstracto de capa por sus siglas en inglés). El SAL es un mecanismo de intercambio y adaptación de datos entre los modelos YANG. Es decir, brinda el servicio de mostrar la infraestructura de los dispositivos y aplicaciones de la red a la capa norte.

Para utilización de la aplicación REST, en este trabajo se empleó el protocolo NETCONF. El cual es un protocolo estandarizado por el IETF (Grupo de Trabajo de Ingeniería de Internet, por sus siglas en inglés), que define un mecanismo simple para gestionar los dispositivos de red. El protocolo utiliza un mecanismo RFC para la comunicación empleando mensajes XML codificados entre un cliente y un servidor. El servidor representa una API que realiza la solicitud al controlador para configurar o conocer el estado de un dispositivo de la red. De esta forma aplicaciones externas pueden acceder al contenido sintáctico como semántico de los dispositivos de la red (RFC 6241, 2011).

## 2.5 Tablas de flujo OpenFlow

OpenFlow es uno de los primeros estándares de facto definidos para SDN. Su objetivo principal es permitir la comunicación directa entre el plano de datos y el controlador, permitiendo que un sistema pueda manejar los cambios de requerimientos constantes. Su arquitectura básicamente es sencilla, un switch o cualquier dispositivo de red que abstraiga su plano de control a través de OpenFlow estará lógicamente compuesto por una serie de tablas de flujo, una tabla grupal y un canal con el que se comunicará con el controlador.

El principal objetivo de las tablas de flujo y la tabla grupal es el de realizar la búsqueda y el reenvío de paquetes. Cada tabla de flujo es un switch que contienen un conjunto de estradas de flujo que a su vez están compuestas por una serie de campos, contadores e instrucciones que se aplicarán a los paquetes que coincidan con las condiciones establecidas a esa entrada de flujo. Es posible que exista más de una tabla de flujo para un switch, en este caso, se establece un orden de prioridad en las tablas. De esta manera se determinan las condiciones de prioridad en las entradas de la tabla de flujos. Si un paquete no encuentra las condiciones necesarias para su redireccionamiento en las tablas de flujo es enviado al controlador o simplemente borrado.

Las entradas de flujo de una tabla de flujo en general redireccionan las salidas a un puerto. Este puerto generalmente es un puerto físico, pero también puede ser uno virtual o reservado. Los puertos reservados pueden realizar acciones específicas, por ejemplo, el reenvío del paquete al controlador, reenvío del paquete mediante un protocolo distinto o una inundación de la red con el paquete (broadcast). De igual forma, los puertos virtuales pueden especificar grupos de agregación de enlaces, túneles o interfaces de loopback.

Las tablas de flujo determinan el direccionamiento de los flujos, generalmente están compuestas por tres componentes fundamentales:

Campo de coincidencia	Contador	instrucción
-----------------------	----------	-------------

**Figura 6.** Estructura de una entra de flujo.

- Campo de coincidencia, son los campos donde se encuentran coincidencias en las tablas de flujo. Suelen ser tanto las cabeceras del paquete como los puertos de entrada a un switch o las direcciones MAC (Media Access control) de destino de un paquete.
- Contador, como su nombre lo indica es un contador que se actualiza únicamente cuando se ha encontrado una coincidencia entre un paquete y la entrada de una tabla de flujo.
- Instrucciones, son las acciones que indican los procedimientos que se han de aplicar en los paquetes una vez hayan encontrado coincidencias en las tablas de flujo.

## 2.6 Calidad de servicio

Por definición de la ITU, Calidad de servicio de telecomunicaciones es: “La totalidad de las características de un servicio de telecomunicaciones que determinan su capacidad para satisfacer las necesidades explícitas e implícitas del usuario o servicio” (UIT-T, 2008).

Los administradores de red utilizan la calidad de servicio (QoS), para garantizar el rendimiento de las aplicaciones que requieran un trato preferencial y garanticen las condiciones requeridas. Se utiliza QoS para gestionar la prioridad y recursos de la red, de modo que tengan los recursos de red necesarios. No todas las aplicaciones requieren las mismas condiciones, por ejemplo, la transmisión de un documento electrónico no puede tolerar la pérdida de paquetes porque el destinatario sería incapaz de reconstruir el archivo por los paquetes faltantes, en cambio una llamada de VoIP puede tolerar cierta pérdida de paquetes, la calidad de la llamada es afectada pero la voz es comprensible, sin embargo, la latencia puede entrecortar la llamada haciendo incomprensible el mensaje que se desea transmitir, mientras que el correo electrónico no presentaría ningún inconveniente por la latencia.

El objetivo de la calidad de servicio es proporcionar el servicio de entrega preferencial para las aplicaciones que lo requieran, garantizando suficientes recursos, el control de retardos y fluctuaciones, además de la reducción en la pérdida de datos. Para medir la calidad de servicio en un entorno de red, se toman en cuenta los siguientes parámetros tradicionales como métricas formales para su medición:

- Disponibilidad del servicio (Availability): La fiabilidad de la conexión de los usuarios. Número de intentos exitosos entre el número total de intentos; medido en porcentaje.
- Retardo (Delay): El tiempo que tarda un paquete para viajar a través de la red desde un extremo a otro; medido en tiempo.
- Fluctuación del retardo (Jitter): La variabilidad del retardo en ejecución entre los paquetes a través de la red desde un extremo a otro; medido en tiempo.
- Caudal eficaz (Throughput): La tasa de transmisión real en el transporte de datos a través de la red (sin encabezados, sin tiempos de guarda, etc.) y siempre es inferior a la tasa de transmisión; medido en bits por segundo.
- Tasa de transmisión (Bit rate): La velocidad máxima de la transmisión total, este término también se utiliza para nombrar a los recursos de velocidad máxima del canal o el servicio; medido en bits por segundo.
- Tasa de pérdida paquetes (Packet loss): La cantidad de paquetes que son descartados, perdidos o dañados mientras atraviesan la red y no llegan al otro extremo; este parámetro es medido en porcentaje.
- Latencia (Latency): Se define como el tiempo que tarda un paquete en ser procesado dentro de un dispositivo, es decir, el tiempo entre el arribo de un paquete, su procesamiento dentro del dispositivo y una vez que sale del dispositivo; medido en tiempo.

### **2.6.1 Definición de parámetros de QoS en la red**

Como uno de los principales objetivos de este trabajo es el enrutamiento multiobjetivo cumpliendo con los parámetros de calidad de servicio para la aplicación. Se consideraron tres parámetros como métricas QoS y medir su rendimiento en el proceso de enrutamiento, estas son: retardo, fluctuación del retardo y tasa de transmisión. A continuación, se describe como se calcula cada uno de ellos (Raayatpanah et al., 2014).

## 2.6.2. Tasa de transmisión

La tasa de transmisión corresponde al flujo de información consumida en el trayecto de un enlace, el cual se puede definir como la ecuación (1).

$$x_e^{m,k} \leq u_e \quad (1)$$

Donde:

X: flujo de información

e: enlace

m: sesión entre nodo - destino

k: nodo destino  $k \in T^m$ .

$T^m$ : destino en sesión m.

$u_e$ = capacidad del enlace

En caso de múltiples sesiones en el mismo enlace, la suma de las velocidades de transmisión consumidas en el enlace "e" por las sesiones "m" hacia cada destino "k", tiene que ser menor o igual a la capacidad total del enlace "e" (ecuación (2)).

$$\sum_{m \in T^m} (x_e^{m,k}) \leq u_e \quad (2)$$

En la comunicación de una trayectoria definida, la velocidad máxima de transmisión ( $x_e^{m,k}$ ), desde el nodo de origen ( $s^m$ ) hasta el nodo de destino ( $T^m$ ) a través de nodos intermediarios (k nodos), estará dada por la capacidad máxima disponible del enlace con el menor valor en la trayectoria establecida, ecuación (3).

$$\max(x_e^{m,k}) \leq \min(u_e) \quad (3)$$

En la figura 7, se muestra un ejemplo de un enlace. En el cual para determinar la velocidad máxima de transmisión  $x_e^{m,k}$ , se tienen las siguientes capacidades:  $u_{e1}=8\text{Gbps}$ ,  $u_{e2}=10\text{Gbps}$  y  $u_{e3}=2\text{Gbps}$ , siendo determinada la máxima tasa de transmisión en la sesión (m) por el mínimo valor de capacidad disponible  $e^x$ , por lo tanto, la máxima la tasa de transmisión empleando la ecuación 3 en el ejemplo será 2Gbps.

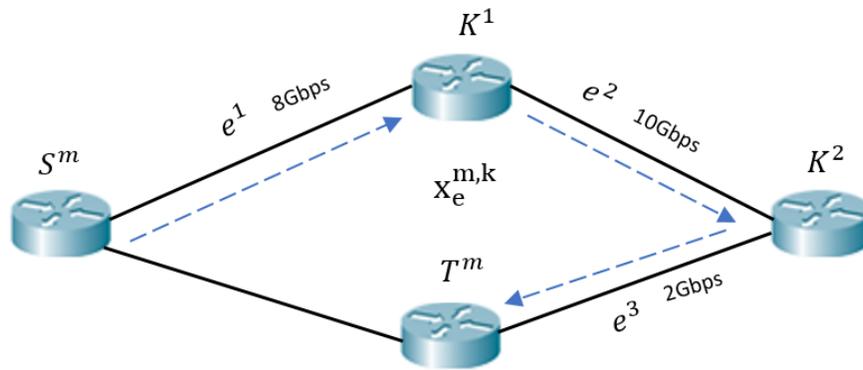


Figura 7. Máxima tasa de transmisión en sesión.

### 2.6.3. Retardo

Considerando una trayectoria de comunicación estable de extremo a extremo, el retardo total es representado por  $D^{m,k}(p)$ , donde una trayectoria "p" hacia su destino "k", se define como la suma de los retardos ( $d_e$ ) de cada enlace más la suma de las latencias ( $d_k$ ) de cada uno de los dispositivos pertenecientes a la trayectoria "p" (ecuación 4). De manera general se puede representar como la sumatoria de retardos y latencias establecidas en la trayectoria de un nodo de origen a su destino como lo representa la ecuación 5.

$$D^{m,k}(p) = (d_{e1}) + (d_{e2}) \dots (d_{en}) + (l_{k1}) + (l_{k2}) \dots (d_{km}) \quad (4)$$

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

La probabilidad del retardo de una trayectoria sea igual o menor a la máxima tolerancia de retardo en un enlace, es igual a uno menos la probabilidad de violación de la restricción de retardo " $\beta^{m,k}$ ", desde el nodo de origen al de destino, como se re representa en la ecuación 6.

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

Dado que la variable aleatoria " $D^{m,k}(p)$ ", toma solo valores positivos con una media finita, permite que el acotamiento sobre la probabilidad de retardo extremo a extremo puede ser obtenido usando las desigualdades de Markov's en la ecuación 6, teniendo como resultado la ecuación 7.

$$Pr(D^{m,k}(p) \leq D_{max}^{m,k}) \leq \frac{E(D^{m,k}(p))}{D_{max}^{m,k}} \quad (7)$$

Para garantizar la probabilidad de violación de extremo a extremo, se emplear la restricción de la ecuación (6) en cada camino " $p \in \mathcal{P}^{m,k}$ ", representado en la ecuación 8.

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

El promedio de retardo extremo a extremo se puede representar como la suma de los retardos promedio en cada enlace "e", más la suma de las latencias promedio en cada nodo del camino "p", representado en la ecuación 9.

$$E(D^{m,k}(p)) = \sum_{e \in p} \bar{d}_e + \sum_{k \in p} \bar{l}_k \quad (9)$$

Si la trayectoria "p" es usada para transmitir un flujo hacia el nodo "k", entonces la ecuación (10), satisface la desigualdad.

$$\frac{\sum_{e \in p} \bar{d}_e + \sum_{k \in p} \bar{l}_k}{D_{max}^{m,k}} \leq \beta^{m,k} \quad (10)$$

Donde el retardo en la trayectoria "p" se define como la restricción de flujo de la información  $F(p)$ , como se representa en la ecuación 11.

$$D(p) = \begin{cases} \frac{\sum_{e \in p} \bar{d}_e + \sum_{k \in p} \bar{l}_k}{D_{max}^{m,k}} & F(p) > 0 \\ 0 & \text{otros casos} \end{cases} \quad (11)$$

## Capítulo 3. Optimización multiobjetivo y algoritmos genéticos

---

### 3.1 Introducción

Cuando en un problema se tienen varias funciones objetivo, la tarea de encontrar una o más soluciones óptimas se denomina optimización multiobjetivo o toma de decisiones multicriterio. Donde los métodos de búsqueda comunes para encontrar la mejor solución posible pueden ser tan simples como una solución aleatoria o tan costosos como hacer una búsqueda exhaustiva en todo el espacio de decisión, evaluando cada posible solución, realizando un ranking que permita seleccionar la solución con un mejor rendimiento. Sin embargo, para entornos de búsqueda muy grandes resulta ser muy ineficiente, o imposible de solucionar en algunos de los casos. Bajo este enfoque, la atención no se puede centrar en un método exhaustivo o en un método que considere únicamente un objetivo, diferentes soluciones pueden producir escenarios en conflicto entre diferentes objetivos; una solución que es óptima con respecto a un objetivo puede no serlo para el resto, por lo tanto, es impropio escoger dicha solución como óptima al problema, por lo cual se crea la necesidad de establecer un compromiso entre los objetivos.

El problema multiobjetivo se presenta en este trabajo al considerar rutas seguras que satisfagan las necesidades de calidad de servicio de los diversos flujos de las aplicaciones, teniendo en conflicto la seguridad y las diversas métricas de QoS como el ancho de banda (bandwidth), retardos (delay y fluctuaciones) y pérdidas de paquetes. Además de la necesidad de encontrar una solución aceptable en un tiempo aceptable. Siendo la principal motivación para el uso de novedosos algoritmos inteligentes, basando la búsqueda en el conocimiento previo del problema (heurísticas) y la utilización de diversos métodos estocásticos, como son los algoritmos evolutivos y genéticos por su bajo consumo de recursos computacionales y velocidad de respuesta aceptable, que utilizan una función de aptitud o fitness, para asignar un valor a cada posible solución y una codificación o mapeo entre el espacio de solución y el algoritmo. Este tipo de técnicas puede lograr alcanzar el caso óptimo para un problema dado, sin embargo, en la mayoría de los casos no se puede garantizar que en todas las ejecuciones se obtenga la solución óptima, pero en general las soluciones resultantes son consideradas “aceptables” en comparación al esfuerzo que debe realizar un método determinado para la búsqueda de una ruta óptima. Siendo un objetivo crucial para establecer el servicio de enrutamiento seguro con calidad de servicio.

### 3.2 Características de la optimización multiobjetivo

El tratado de problemas de optimización que consideran varios objetivos comenzó con el trabajo seminal de Pareto (1896). Representando de forma general el problema de optimización multiobjetivo, como se representa en la ecuación (12).

$$\frac{\min}{\max}: f_m(x), m = 1, 2, \dots, m \quad \text{ec. (12)}$$

$$g_j(x) \geq 0, j = 1, 2, \dots, J \quad \text{ec. (13)}$$

$$h_k(x) = 0, h = 1, 2, \dots, K \quad \text{ec. (14)}$$

$$x_i^j \leq x \leq x_i^u, i = 1, 2, \dots, N \quad \text{ec. (15)}$$

Para resolver el problema es necesario definir criterios con el fin de determinar cuáles soluciones se consideran factibles y cuáles no. Se emplea el concepto de dominancia, que conlleva al proceso de clasificar las diferentes soluciones y de encontrar las alternativas más factibles, teniendo en cuenta la presencia y cuantificación de los M objetivos del problema (ec. 13 – ec. 15).

Se dice que una solución  $x_1$  domina a otra solución  $x_2$  si se cumplen las siguientes condiciones:

- La solución  $x_1$  no es menor a  $x_2$  en todos los objetivos.
- La solución  $x_1$  es estrictamente mejor que  $x_2$ , por lo menos en un objetivo.

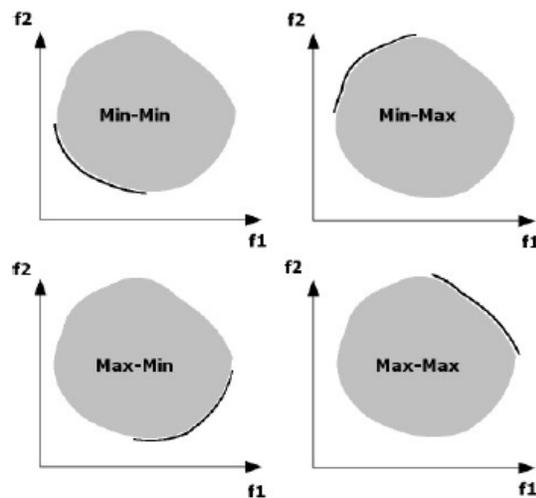
Si alguna de las condiciones es violada, la solución  $x_1$  no domina la solución  $x_2$ .

El concepto de dominancia se puede extender para encontrar un conjunto de soluciones no dominadas dentro de una población. Si se considera una población de N soluciones cada una con M valores de funciones objetivo. Se utiliza el siguiente procedimiento para encontrar el conjunto no dominado de soluciones:

1. Hacer  $i = 1$ .
2. Para todos lo  $j \neq i$ , comparar las soluciones  $x_i$  y  $x_j$  para determinar dominancia, utilizando las dos condiciones de dominancia para todos los M objetos.
3. Si para algún  $j$ ,  $x_i$  es dominado por  $x_j$ , marcar  $x_i$  como dominado. Incrementar  $i$  en uno y regresar al paso 2.
4. Si todas las soluciones ( $i=j$ ) en el conjunto pasar al paso 5, de lo contrario incrementar  $i$  en uno y regresar al paso 2.
5. Todas las soluciones que nos marcadas como dominadas son soluciones no-dominadas.

Cuando se realiza un análisis de dominancia entre dos soluciones y se encuentra que la primera condición de dominancia no se cumple para ninguna de las dos condiciones, no se puede concluir acerca de la dominancia de una con respecto a la otra. Cuando esto ocurre, se dice que las soluciones son no-dominadas. De esta manera, se tiene un conjunto finito de soluciones y se realiza una comparación de todos los pares posibles; al final se obtiene un conjunto de soluciones que no son dominadas entre sí y dicho conjunto tiene la propiedad de dominar al resto de soluciones que no pertenezcan a él. Por lo tanto, se tiene un conjunto de soluciones "p", el conjunto no dominado de soluciones "p'" lo forman aquellas que no son dominadas por ningún miembro del conjunto "P", denominado a este conjunto frente de Pareto (Coello et al., 2007).

En la figura 8, se muestran los conjuntos óptimos de Pareto para diferentes escenarios con dos objetivos en el mismo espacio de soluciones. En cualquiera de los casos el frente de Pareto está compuesto por soluciones ubicadas en el borde de la región factible del espacio de soluciones.



**Figura 8.** Frentes óptimos de Pareto en el mismo espacio de soluciones.

Una población puede ser clasificada en diferentes niveles de no dominancia. Cuando el procedimiento de los cinco pasos para encontrar el conjunto no dominado es aplicado por primera vez, el conjunto resultante es el conjunto no dominado de mejor nivel (primer frente de Pareto). Para obtener más clasificaciones, se omiten los elementos de esta solución y se repite el mismo procedimiento para obtener el segundo nivel de Pareto. Este proceso puede se puede repetir hasta que todos los elementos de la población sean clasificados en un nivel.

Uno de los objetivos de la optimización para problemas multiobjetivo, es encontrar tantas soluciones factibles, como sea posible. Dado que los algoritmos evolutivos utilizan una población de soluciones, se pueden encontrar soluciones no dominadas óptimas.

### 3.3 Algoritmos genéticos

La computación evolutiva es un área de la informática que se inspira en el proceso de evolución natural de las especies, donde cada una de ellas sobrevive adaptándose a su medio. Considerando la evolución simplemente como un proceso en cual, una población de individuos lucha solo por su sobrevivencia y reproducción donde la condición física de los individuos resulta crucial para su adaptación y éxito. Representando en otro contexto, es un problema en el cual un conjunto de soluciones candidatas, pueden resolver un problema dependiendo de sus cualidades, con la posibilidad de construir una mejor solución a partir de soluciones potenciales. Este estilo de resolución de problemas se basa en el contexto de prueba y error hasta llegar a una condición de parada. Este principio es basado en la metaheurística de población descrito en el algoritmo 1. Primero se aplica una búsqueda local a una solución inicial, luego, por cada interacción se realiza una perturbación de cada óptimo local obtenido. Finalmente, se realiza una búsqueda local a cada solución perturbada. Repitiéndose este proceso hasta llegar a la condición de parada (Eiben y Smith, 2015).

Los principios de adaptación y evolución de sistemas biológicos son atribuidos a Darwin, 1987. Siendo, implementado por primera vez para construir maquinas inteligentes por Alan Turing en 1948, y para 1962 Bremermann, ejecutó experimentos computacionales sobre optimización a través de la evolución y recombinación. Siendo la inspiración conceptual que les permitió a Rechenberg (1965), Fogel et. Al., (1966), Holland (1975) y Koza (1992), desarrollar sus propias propuestas de estrategias evolutivas y genéticas. Esta familia de algoritmos conforma la base de lo que hoy en día se conoce con el nombre de cómputo evolutivo (CE), (Kennedy & Eberhart 2001; de Castro & Timmis, 2002; Olague et, al., 2006).

Los algoritmos genéticos (GA por sus siglas en inglés), fueron propuestos por Holland (1975), mientras que el trabajo de Goldberg (1989), ayudó a implementarlos como una herramienta robusta y flexible para los problemas de “búsqueda”, “optimización” y “aprendizaje de máquinas”. Esta familia de algoritmos no solo se basa en el marco conceptual de supervivencia del más apto de Darwin, también incorpora las teorías de herencia de rasgos a través de los genes descrita por Mendel.

---

**Algoritmo 1** Pseudocódigo de metaheurística basada en población
 

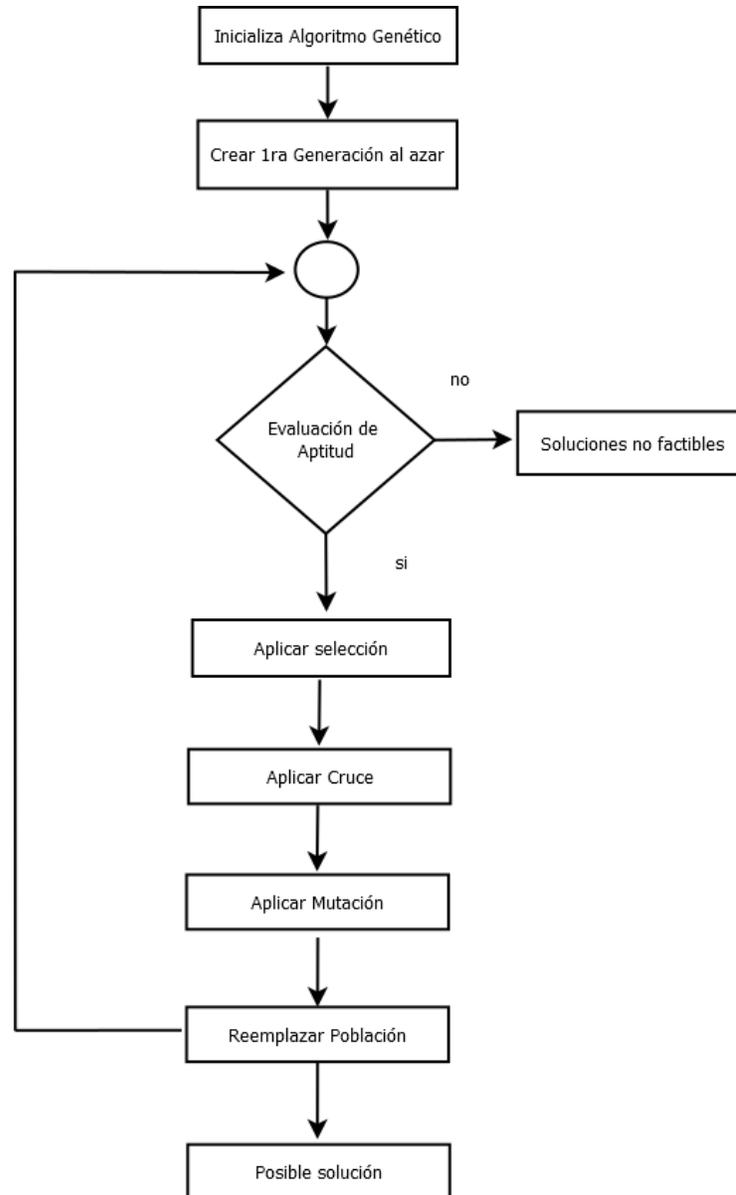
---

1.  $P = P_0$  // Generación de la población inicial
2.  $T=0$ ;
3. **While** //Condición de terminación
4. Generar ( $P'_t$ ); //Generación de la nueva población
5.  $P_{t+1}$ = Seleccionar Población  $P_t$  U  $P'_t$  // selección de la nueva población
6.  $t = t + 1$  ;
7. **parar** //Criterio de parada
8. **Output** Mejores soluciones encontradas

### 3.3.1 Características de los algoritmos genéticos

Dado un problema específico a resolver con GA evolutivos, es necesario tener primero un conjunto de posibles soluciones potenciales a ese problema conocido como una población, la cual representa un conjunto de individuos de la misma especie denominado multiconjunto de cromosomas. Los cromosomas o posibles soluciones requieren ser codificados a través de una métrica llamada función de aptitud. Se permite evaluar cuantitativamente a cada una de las soluciones denominadas individuos, estas no necesariamente representan una solución óptima por lo cual es necesario mejorarlas con el proceso evolutivos y adaptativos para lograr encontrar mejores soluciones como se muestra en la figura 9.

Las soluciones que no resultan ser prometedoras son eliminadas desacuerdo con la función de aptitud. Sin embargo, por azar algunas pueden ser prometedoras, mostrando una aptitud débil que puede contribuir a una solución factible. Las soluciones prometedoras se conservan y son empleadas para reproducirse (cruzamiento) con otros individuos. Se realizan múltiples copias de ellas, las cuales se introducen a cambios aleatorios (mutaciones) durante el proceso de copiado. Esta descendencia prosigue con la siguiente generación, formando un nuevo conjunto de soluciones candidatas y son sometidas a una ronda de evaluación de aptitud. Sí, las soluciones presentan una aptitud más baja o ninguna mejora aparente después del proceso evolutivo son eliminadas; mientras las que representen una mejora, son soluciones más eficientes al problema. Este proceso se repite para crear nuevas generaciones de individuos, el cual termina al no encontrar mejoramiento en la población o después de un número predeterminado de repeticiones.



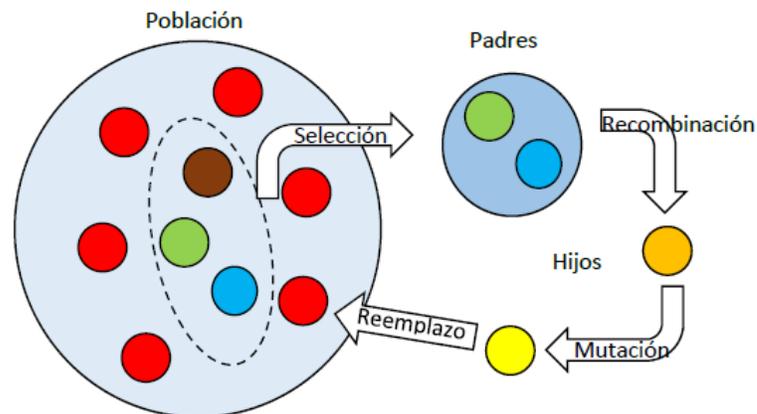
**Figura 9.** Diagrama de algoritmo evolutivo simple.

En cuanto a la representación genética de GA, un individuo es representado por una cadena de elementos denominada cromosoma, donde cada cromosoma está compuesto por una serie de genes que determinan sus rasgos o propiedades particulares. Cuando se habla de esta forma de representación, es necesario identificar:

- **Espacio de genotipo:** En este espacio están los cromosomas y sus genes correspondientes, que codifican las características de un individuo. También, los procesos de variación u operadores trabajan en este espacio, como el cruce (combinación) y la mutación.

- Espacio de fenotipo: este es el espacio del individuo, siendo la solución, una vez que es decodificada a partir de su cromosoma.
- Espacio de aptitud: Este es el espacio de la función de aptitud, donde suelen operar los procesos de selección y supervivencia.

En el GA canónico, los cromosomas son cadenas de bits de longitud fija. Estos GA emplean dos tipos de operadores básicos para generar individuos nuevos (hijos), utilizando los individuos existentes (padres) dentro de la población como se muestra en la figura 10. El cruce o recombinación, toma los cromosomas de dos individuos padres y a partir de estos construye un individuo hijo. Lo que se espera, es que los padres con una aptitud alta transmitan los genes necesarios al hijo para obtener una aptitud aún mayor al de ellos. De esta manera se pretende construir soluciones cada vez más aptas; estas son las bases de la llamada teoría de Schema o patrones (Holland, 1967; Golber, 1989). Mientras la mutación, se aplica como un proceso aleatorio, modifica el cromosoma de algunos de los individuos con el propósito de producir genes que no existían en ningún individuo de la población. Funcionando como un mecanismo de ayuda para los GA de salir de los óptimos locales. En el algoritmo 2 se representa el pseudocódigo de un AG canónico.



**Figura 10.** Representación gráfica del ciclo evolutivo para un GA.

---

**Algoritmo 2** Pseudocódigo de un AG

---

1. */\*Parametros de un AG\*/*
2.  $t = 0$  // número de generación
3.  $P = P_0$  // Generación de la población inicial
4. Evaluación  $P_0$ ;
5. **While** //Condición de terminación
6.  $P'_t$ ; //Selección de la población
7.  $P'_t =$  aplicar operadores de cruce  $P_t$
8.  $P'_t =$  aplicar operadores de mutación  $P_t$
9.  $P_{t+1} =$  Reemplazo de Población  $P_t$  U  $P'_t$  // selección de la nueva población
10. Evaluar  $P'_{t+1}$
11.  $t = t + 1$  ;
12. **Parar** // Criterio de parada
13. **fin** Mejores soluciones encontradas

**3.3.2 Métodos de selección**

Existen diferentes técnicas para seleccionar a los individuos que deben copiarse hacia la siguiente generación. Algunos de estos métodos son mutuamente exclusivos, pero otros pueden utilizarse en combinación. Los más empleados son:

- Elitista: Se garantiza la selección de los miembros más aptos de cada generación. Suele copiarse el mejor o mejores individuos hacia la siguiente generación en caso de que no surja un individuo mejor.
- Proporcional a la aptitud: los individuos más aptos tienen más probabilidades de ser seleccionados, pero no existe certeza. Otorgándole una mayor de probabilidad de selección a los individuos más aptos.
- Ruleta: una forma de selección proporcional a la aptitud en la que la probabilidad de que un individuo sea seleccionado es proporcional a la diferencia entre aptitud y la de sus competidores.
- Torneo: se eligen subgrupos de individuos de la población, y los miembros de cada subgrupo compiten entre ellos. Solo se elige a un individuo de cada subgrupo para la reproducción.

- Por rango: a cada individuo de la población se le asigna un rango numérico basado en su aptitud, y la selección se basa en esta asignación, en lugar de las diferencias absolutas en aptitud. La ventaja de este método es que puede evitar que individuos muy aptos ganen dominancia de los menos aptos.
- Generacional: sustituye todos los individuos de una generación para obtener la siguiente.

### 3.3.3 Métodos de cruzamiento

El cruzamiento implica seleccionar a dos individuos para que intercambien segmentos de su código, produciendo una descendencia cuyos individuos son combinaciones de sus padres. Las formas comunes de cruzamiento son en un punto, dos puntos y uniforme. Para ilustrarlas, supondremos que tenemos las soluciones de un problema particular de tipo binario con una cadena de ocho elementos. Los individuos A y B han sido seleccionados para ser sometidos a los métodos de cruce mencionados:

Individuo A: 11101001

Individuo B: 01001100

- Operador de cruce simple o de un punto: como su nombre lo indica, se establece un punto de intercambio en un lugar aleatorio del genoma de los dos individuos. El individuo A contribuye todo su código anterior a ese punto y el individuo B contribuye todo su código a partir de ese punto para producir una descendencia, como se muestra en la figura 11.

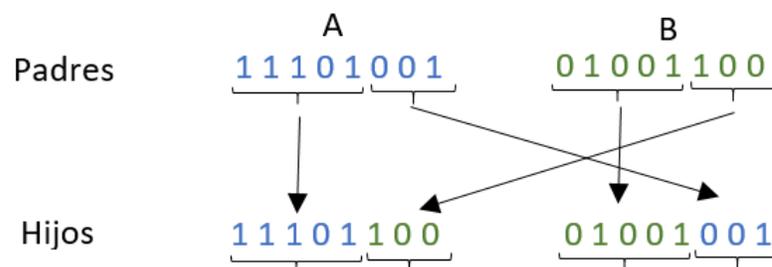


Figura 11. Operador de cruce simple.

- Operador con dos puntos de corte: es análogo al anterior, salvo que se establecen dos puntos de intercambio en dos lugares aleatorios del genotipo de los individuos, intercambiando sus códigos para producir dos hijos. La figura 12 muestra un ejemplo de este operador.

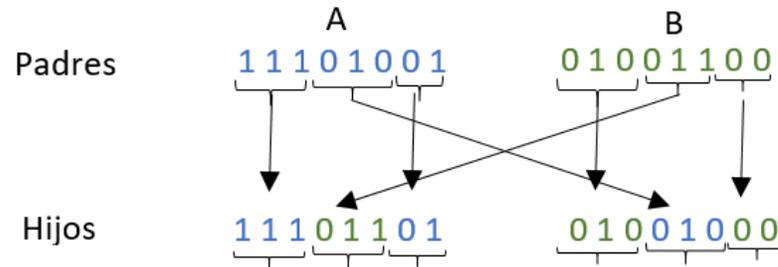


Figura 12. Operador con dos cruces de corte.

- Operador de cruce uniforme: los genes de los descendientes se obtienen de acuerdo con el criterio dado por una máscara. La máscara se puede generar como un individuo aleatorio de unos y ceros, y se intercambian los bits de los dos cromosomas que coincidan donde hay un 1 en el individuo. La figura 13 muestra ilustra un ejemplo de este operador.

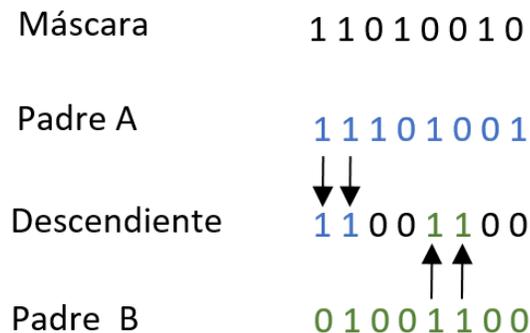
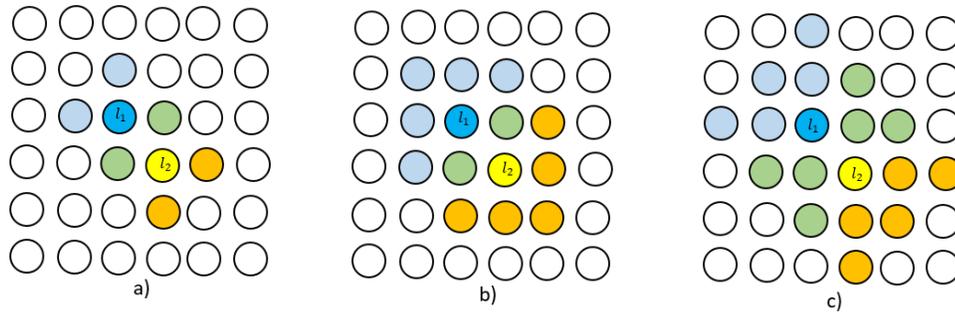


Figura 13. Operador de cruce uniforme.

### 3.3.4 Métodos de mutación

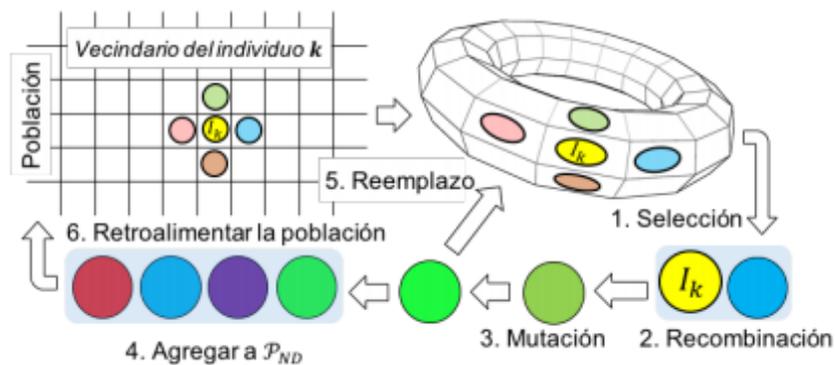
La mutación es un operador genético utilizado para proporcionar diversidad en las generaciones precedentes de una población. Altera el estado de una secuencia genética de un individuo al intercambiar un elemento de su cadena, teniendo como propósito evitar los mínimos locales (poblaciones de individuos muy similares entre sí), los cuales vuelven lento el proceso evolutivo o en su defecto lo pueden detener. Por ejemplo, continuando con el descendiente de la figura 11, la mutación consistiría en intercambiar uno





**Figura 14.** Vecindarios, a) cuatro, b) ocho, c) doce. Para los individuos  $I_1=(x,y)$  e  $I_2=(x-1,y-1)$ .

Mocell construye un conjunto ( $P_{ND}$ ) de  $N$  individuos no dominados en donde cada individuo es evaluado a través de un torneo entre vecinos para determinar los padres o semillas de una generación de la población, los cuales se utilizarán después en el proceso de recombinación (cruzamiento), para luego emplear el operador de mutación con los hijos resultantes, continuando con una nueva evaluación de ranking con el fin de determinar el mejor rendimiento de los individuos. Por último, al finalizar la revisión de todos los individuos, de manera aleatoria se seleccionan  $N$  individuos que serán intercambiados por lo que se encuentran alojados en el conjunto ( $P_{ND}$ ). La figura 15 y el pseudocódigo 3 representan e ilustran el funcionamiento del algoritmo de Mocell.



**Figura 15.** Representación del algoritmo genético Mocell con un conjunto de cuatro individuos (Peña, 2017).

---

**Algoritmo 3** Pseudocódigo de algoritmo *moCell*

---

1.  $data = \text{setup}()$ ; /\*Configuración de parámetros del algoritmo y datos de entrada\*/
2.  $pop = \text{popGen}()$ ; /\*Crea la población inicial\*/
3.  $pop = \text{toroidal}(pop)$ ; /\*Distribuye los individuos en una malla toroidal\*/
4.  $p_{ND} = []$ ; /\*Crea un conjunto para almacenar los individuos no dominados\*/
5. **While** (condición Determinación==*falso*) **hacer**
6. **Para**  $k = 1$  **hasta**  $|pop|$  **hacer**
7.  $l_k = pop(k)$ ; /\*individuo  $k$ \*/
8.  $v_{pop} = \text{vecindario}(pop, l_k)$ ;
9.  $padres = \text{selección}(vpop)$ ;
10.  $hijos = \text{recombinación}(data, padres)$ ;
11.  $hijos = \text{mutación}(data, hijos)$ ;
12.  $pop(k) = \text{reemplazo}(vpop, hijos)$ ;
13.  $p_{ND} = \text{agregar}(pop(k), p_{ND})$ ;
14. **Fin**
15.  $pop = \text{retroalimentación}(p_{ND})$ ;
16. **Fin**

## Capítulo 4. Propuesta de enrutamiento con algoritmo genético (mocell)

---

### 4.1 Introducción

En la actualidad el enrutamiento es uno de los problemas más conflictivos y latentes en las redes de telecomunicaciones, principalmente por el crecimiento acelerado de dispositivos y la demanda constante de servicios. Las solicitudes de servicios se vuelven cada vez más difíciles de atender por la diversidad de requerimientos que se deben administrar, estos se encuentran relacionados con la calidad de servicio y la gestión de recursos como: ancho de banda, retardos, pérdidas de datos, confiabilidad, entre otras, para el cálculo de una adecuada trayectoria.

En las redes convencionales como SDN existen una gran diversidad de protocolos de enrutamiento, algunos de ellos orientados a QoS; sin embargo, la mayoría de ellos suelen considerar únicamente una métrica para el proceso de enrutamiento proporcionando trayectorias concurridas y el constante riesgo de producir congestiones en la red o el mal funcionamiento o rendimiento de una aplicación resultando necesario la implementación de múltiples métricas para mejorar el proceso de enrutamiento y proporcionar rutas más robustas con la capacidad de adaptación a los cambios constantes de la red. En este capítulo, se describe la incorporación del algoritmo genético mocell (descrito en el capítulo 3) en el proceso de enrutamiento con múltiples métricas en una red con dos dominios, describiendo el proceso particular de codificación y generación de la población inicial, comprendida por las trayectorias entre los nodos de los extremos. Finalizando con la función fitness y la selección de la ruta con la mayor capacidad para transferir información en el proceso de emulación la red SDN y controladores.

### 4.2 Implementación del algoritmo genético

Como se mencionó en el capítulo 3, los algoritmos genéticos parten de un conjunto (población) de soluciones potenciales (individuos) para el problema. A las cuales se les aplican operadores de mutaciones y recombinación para encontrar mejores soluciones al problema. En nuestro caso, es necesario proporcionar rutas que establezcan la comunicación entre un nodo de origen y uno de destino. La forma más sencilla para generar rutas es a partir de la búsqueda aleatoria. Para ello, se emplean arreglos de números enteros aleatorios para simbolizar diversas rutas.

### 4.2.1 Representación de codificación propuesta para enrutamiento

Para la generación de una ruta codificada simplemente se crea una cadena que contenga el número total de dispositivos de la red ordenados aleatoriamente. Las posiciones de los números representan los genes (IDs orden de prioridad en los saltos) que conforman un cromosoma (ruta). Donde el primer gen representa el origen del enrutamiento y para determinar la ruta de un cromosoma es necesario decodificarla empleando el nodo de destino y la tabla de adyacencias de la topología. Por ejemplo, para realizar la codificación de una ruta entre H1 y R1 de la red propuesta en la figura 16 se realiza el siguiente procedimiento:

#### codificación

- Se genera una cadena con 10 elementos correspondientes a los dispositivos de la red.
- La cadena es llenada con los números aleatorios del 1 al 10.
- Se crea un cromosoma: **5 8 2 1 4 10 6 7 3**

#### Decodificación de la ruta

- Se emplea la tabla de adyacencias de la red para realizar este procedimiento y el valor de los genes de mayor a menor para determinar el orden de la ruta como se muestra a continuación:
  1. Inicializa la ruta en H1, como R4 es el único nodo adyacente se realiza el salto H1 -> R4
  2. R4 cuenta con dos nodos adyacentes R2 con un ID= 8 y R5 con un ID=4. Se selecciona el nodo con el mayor ID, valor de por lo cual se toma la ruta hacia R2.
  3. En R2 se elige R1 por tener el ID con mayor valor.
  4. Se concluye el proceso de decodificación, la ruta alcanzo el nodo final propuesto.
- Ruta: **H1 -> R4 -> R2 -> R1**

El pseudocódigo del algoritmo 4 representa el procedimiento de codificación para el proceso de enrutamiento del algoritmo genético propuesto en este trabajo. En cuanto a los operados genéticos se emplearon únicamente el operador de cruce simple para el cruzamiento de genes y el método de mutación aleatorio en un gen.

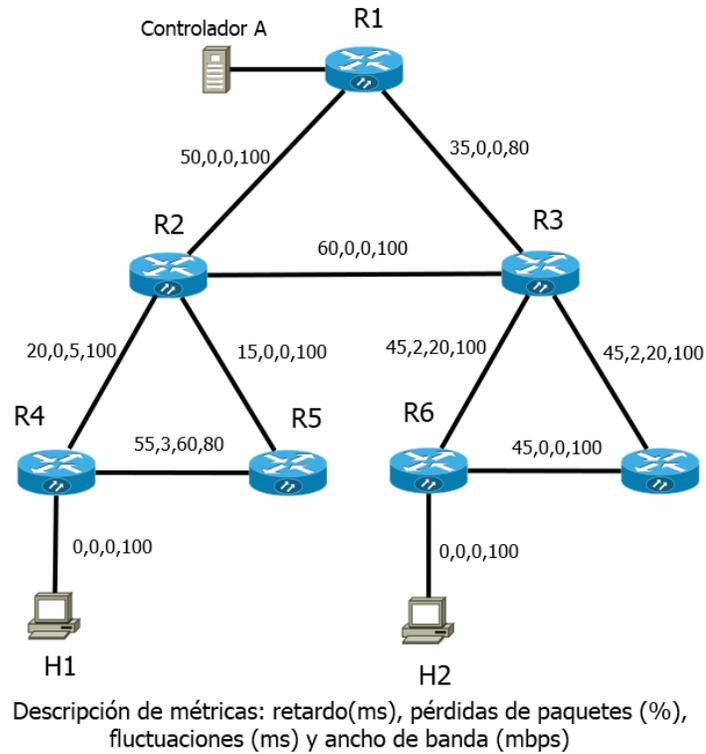


Figura 16. Topología propuesta para el ejemplo de representación de codificación.

Tabla. 1 . Tabla de adyacencias del controlador A para la representación de codificación.

Cromosoma	5	8	2	1	4	10	6	7	3	9
Nodos	R1	R2	R3	R4	R5	R6	R7	H1	H2	H3
R1	0	1	1	0	0	0	0	0	0	0
R2	1	0	1	1	1	0	0	0	0	0
R3	1	1	0	0	0	1	1	0	0	0
R4	0	1	0	0	1	0	0	1	0	0
R5	0	1	0	1	0	0	0	0	0	0
R6	0	0	1	0	0	0	1	0	1	0
R7	0	0	1	0	0	1	0	0	0	1
H1	0	0	0	1	0	0	0	0	0	0
H2	0	0	0	0	0	1	0	0	0	0
H3	0	0	0	0	0	0	1	0	0	0

---

**Algoritmo 4** Pseudocódigo de codificación del algoritmo genético

---

1. *data =setup( ); /\*Asignación de ID los nodos \*/*
2. *cromosoma =ID( )/\*Crea un cromosoma aleatorio de acuerdo al tamaño de la red (IDs)\*/*
3. *origen=VS /\*se especifica el nodo de inicio*
4. *destino=Vd /\*se determina el destino fin de la decodificación\*/*
5. *matriz= Adyacencias (data) /\*crea una tabla de adyacencias respecto a la red \*/*

6. *decodificación=adyacencias[vs, vd] /\*decodifica el cromosoma de acuerdo con el orden de prioridad ID en la tabla de adyacencias, iniciando en Vs y finalizando en Vd\*/*
7. *p<sub>ND</sub> = [ ]; /\*Crea un conjunto para almacenar los individuos no dominados (soluciones)\*/*
8. **Fin**

#### 4.2.2 Representación de la función fitness

La topología de la red puede ser representada como un grafo:

$$G(V,E) \quad (16)$$

Donde:

- $V = (1, \dots, m)$  : Representación de nodos.
- $E = (i,j)$  : Conjunto de conexiones (aristas) entre nodos.
- $v_s$  : Nodo de inicio.
- $v_d$  : Nodo de destino.
- $\lambda_{(i,j)}$  : Ancho de banda de la arista.
- $\mu_{(i,j)}$  : Delay de la arista.
- $s_{(i,j)}$  : Nivel de seguridad de la arista.

Para cada ruta es necesario calcular:

- $R_\lambda = \prod_{v_i \in \{v \dots v_n\}} \lambda_{(i,j)}$  : Ruta de acuerdo con el ancho de banda. (17)

- $R_\mu = \prod_{v_i \in \{v \dots v_n\}} \mu_{(i,j)}$  : Ruta de acuerdo con los retardos. (18)

- $R_S = \prod_{v_i \in \{v \dots v_n\}} s_{(i,j)}/j$  : Ruta de acuerdo con el nivel de seguridad. (19)

- 

Teniendo como solución a nuestro problema un grafo donde se tiene que encontrar la ruta con el mínimo retardo y el máximo ancho nivel mientras cumple con un límite en el nivel de seguridad:

$$G \mid \text{minimizar}\{R_\mu\}, \text{maximizar}\{R_\lambda, R_S\} \mid \quad (20)$$

### 4.3 Emulación de la red en Mininet

Mininet (Lantz et al., 2018) es una de las plataformas más comunes y utilizadas en la literatura para emulación de SDN, por su fácil implementación, compatibilidad y flexibilidad con otras aplicaciones y controladores. Este emulador permite realizar la virtualización de una red en un solo núcleo de Linux. Emplea el núcleo y sistema subyacente para ejecutar un conjunto de conmutadores, enrutadores, enlaces y host virtuales y representar su funcionalidad. Un host en mininet se comporta como una máquina virtual real, puede acceder y ejecutar los programas arbitrarios del sistema (programas instalados en el sistema subyacente). Los programas pueden ejecutar el envío de paquetes a través de una interfaz ethernet real, permitiendo su configuración de velocidad y retraso. Los conmutadores, enrutadores o middlebox por su parte, simulando un conjunto de colas similares a las de sistemas reales.

Para la emulación de mininet se utilizó una máquina virtual, con sistema operativo Linux Debian, con 4 vCPUs, 8GB de memoria RAM, 100 GB de espacio en disco (discos NLSAS); con las mismas características se emplearon otras dos máquinas virtuales para la utilización de controladores OpenDaylight.

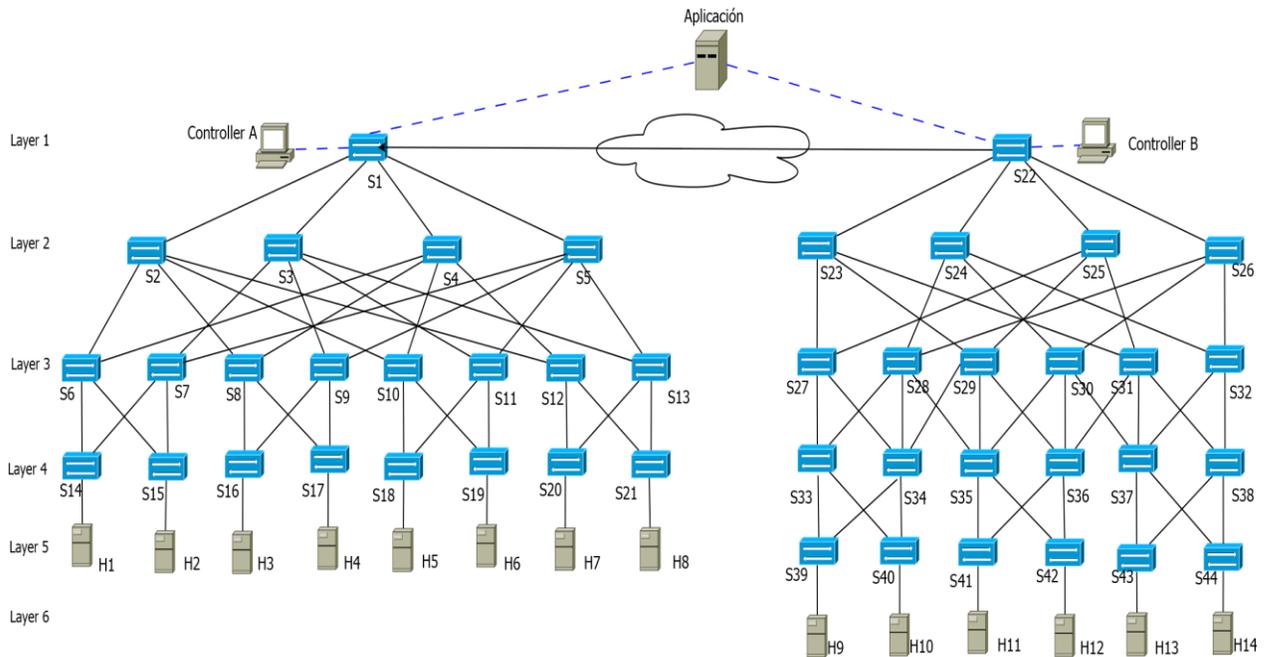
Las versiones de software utilizadas:

- Debian 9
- Mininet 2.3
- OpenDayLight Beryllium (controlador, utilizando REST API como interfaz norte)
- OpenFlow 1.3 (interfaz sur)
- Matlab 2017 (implementación de enrutamiento basado en mocell)
- Iperf (plano de aplicación)

#### 4.3.1 Topología

Para la implementación de enrutamiento empleando el algoritmo genético Mocell, en este trabajo se propone una red distribuida como se ilustra en la figura 17. Estableciendo como condición encontrar la mejor ruta para transferir la mayor cantidad de información a través de los nodos ubicados de extremo a extremo. Es decir, establecer la mejor ruta entre H1 y H14 para transferir información.

La topología está conformada por una red SDN constituida por dos dominios, los cuales se encuentran administrados por los controladores A y B. Como son dominios independientes cada controlador es capaz únicamente de obtener la información y estado de su propio dominio. Por lo consiguiente, es necesario la implementación de una aplicación que coordine ambos controladores para establecer el enrutamiento que involucre la comunicación entre ambos dominios.



**Figura 17.** Topología de red propuesta para el problema de enrutamiento entre extremos.

Los parámetros configurados en cada enlace son retardo, fluctuaciones del retardo y tasa de transmisión, además de proponer un nivel de seguridad cada nodo, este valor puede corresponder a una evolución previa de seguridad a cada nodo mediante un pentesting o test de penetración para evaluar la configuración del dispositivo y determinar sus vulnerabilidades. Considerando para este caso particular un valor aleatorio entre 0 y 5 para determinar un nivel de seguridad en los nodos, donde 0 es un equipo comprometido y 5 es el número más alto de seguridad.

En mininet se asignaron valores predeterminados de retardos, fluctuaciones y capacidad de transmisión por enlace, mientras que el nivel de seguridad se agregó como métrica adicional en el algoritmo de enrutamiento. Los valores de las métricas correspondientes a cada enlace de los controladores A y B de la topología se encuentran representadas en las tablas 2 y 3.

**Tabla. 2.** Métricas de los enlaces correspondientes al controlador A (Delay (ms), Nivel de seguridad, Jitter (ms) y ancho de banda (mbps)).

Enlace	Métricas	Enlace	Métricas	Enlace	Métricas
S1 <-> S2	20,2.2,0,100	S4 <-> S12	20,3.6,0,100	S11 <-> S18	20,2,0,70
S1 <-> S3	15,3.5,0,100	S5<-> S7	20,3.4,10,100	S11 <-> S19	25,2,0,100
S1 <-> S4	10,2.5,0,100	S5 <-> S9	12,4.5,0,100	S12 <-> S20	55,2,0,100
S1 <-> S5	5,3.8,0,100	S5 <-> S11	25,2.7,0,100	S12 <-> S21	15,2,0,80
S2 <-> S6	20,2.3,0,100	S5 <-> S13	15,3.5,0,100	S13 <-> S20	15,2,5,80
S2 <-> S8	15,4.2,0,100	S6 <-> S14	80,2,10,80	S13 <-> S21	20,2,0,100
S2 <-> S10	25,3.8,0,100	S6 <-> S15	60,2,0,60	S14 <-> H1	0,2,0,100
S2 <-> S12	10,3.6,0,100	S7 <-> S14	25,2,0,100	S15 <-> H2	0,2,0,80
S3 <-> S7	15,3.4,5,100	S7 <-> S15	20,2,5,80	S16 <-> H3	0,2,0,70
S3 <-> S9	20,4.5,0,100	S8 <-> S16	20,2,0,100	S17 <-> H4	0,2,0,100
S3 <-> S11	10,2.7,0,100	S8 <-> S17	15,2,0,70	S18 <-> H5	0,2,0,100
S3 <-> S13	20,3.5,5,100	S9 <-> S16	10,2,5,50	S19 <-> H6	0,2,0,70
S4 <-> S6	15,2.3,0,100	S9 <-> S17	15,2,5,80	S20 <-> H7	0,2,0,80
S2 <-> S8	10,4.2,0,100	S10 <-> S18	15,2,0,80	S21 <-> H8	0,2,0,100
S4 <-> S10	15,3.8,0,100	S10 <-> S19	10,2,0,50		

**Tabla. 3.** Métricas de los enlaces correspondientes al controlador B (Delay (ms), Nivel de seguridad, Jitter (ms) y ancho de banda (mbps)).

Enlace	Métricas	Enlace	Métricas	Enlace	Métricas
S22 <-> S23	15,2.5,0,100	S27 <-> S34	15,4.4,5,80	S34 <-> S39	10,2,0,80
S22 <-> S24	10,4.5,0,100	S28 <-> S33	15,4.4,0,100	S34 <-> S40	20,2,0,100
S22 <-> S25	5,3.8,0,100	S28 <-> S34	20,3.8,0,80	S35 <-> S41	15,2,0,100
S22 <-> S26	10,3.6,0,100	S28 <-> S35	15,2.2,10,60	S35 <-> S42	15,2,0,80
S23 <-> S27	15,3.4,0,100	S29 <-> S34	15,3.8,0,70	S36 <-> S41	10,2,0,70
S23 <-> S29	15,2.8,0,100	S29 <-> S35	20,2.2,0,80	S36 <-> S42	15,2,0,100
S23 <-> S31	25,3.6,0,100	S29 <-> S36	20,3.6,0,100	S37 <-> S43	20,2,0,100
S24 <-> S28	10,4.2,0,100	S30 <-> S35	15,2.2,0,70	S37 <-> S44	10,2,0,70
S24 <-> S30	15,4.4,0,100	S30 <-> S36	15,3.6,0,100	S38 <-> S43	15,2,0,80
S24 <-> S32	20,2.8,0,100	S30 <-> S37	15,4,5,80	S38 <-> S43	15,2,0,100
S25 <-> S27	10,3.4,0,100	S31 <-> S36	15,2,5,80	S39 <-> H9	0,020,100
S25 <-> S29	15,2.8,0,100	S31 <-> S37	10,2,0,100	S40 <-> H10	0,2,0,100
S25 <-> S31	10,3.6,0,100	S31 <-> S38	20,2,0,70	S41 <-> H11	0,2,0,100
S26 <-> S28	15,4.2,0,100	S32 <-> S37	20,2,0,100	S42 <-> H12	0,2,0,100
S26 <-> S30	10,4.4,0,100	S32 <-> S38	25,2,0,100	S43 <-> H13	0,2,0,100
S26 <-> S32	20,2.8,0,100	S33 <-> S39	15,2,0,100	S44 <-> H14	0,2,0,100
S27 <-> S33	10,3,5,100	S33 <-> S40	10,2,0,70		

### 4.3.2 Implementación del Controlador OpenDaylight

En este trabajo se empleó el controlador OpenDayLight por su modularidad y flexibilidad para obtener la información del estado de la topología de la red y poder realizar un enrutamiento orientado a la QoS de un flujo, en nuestro caso únicamente se limita en encontrar la ruta que pueda proporcionar la mayor tasa de transferencia, considerando el ancho de banda de los enlaces, retardos y un límite en el nivel de seguridad. Pero antes de entrar en más detalles, es importante mencionar que, la edición del controlador OpenDaylight utiliza el IDE de Maven. Como Maven es una herramienta de automatización alojada en Apache Software Foundation, es necesario estructurar un modelo de objetos orientados a pom.xml, donde se especifican las reglas y acciones que ejecutarán cada uno de los dispositivos de la red.

El procedimiento de enrutamiento se ilustra en la figura 18. La red (figura 17) es emulada por mininet. Los controladores utilizados para cada dominio son ODL. El controlador A pertenece al dominio sodl.cicese.mx. Se encuentra constituida por 29 dispositivos de los cuales 21 son enrutadores y 8 nodos; mientras el controlador B pertenece al dominio sodl2.cicese.mx y cuenta con 23 enrutadores y 6 nodos.

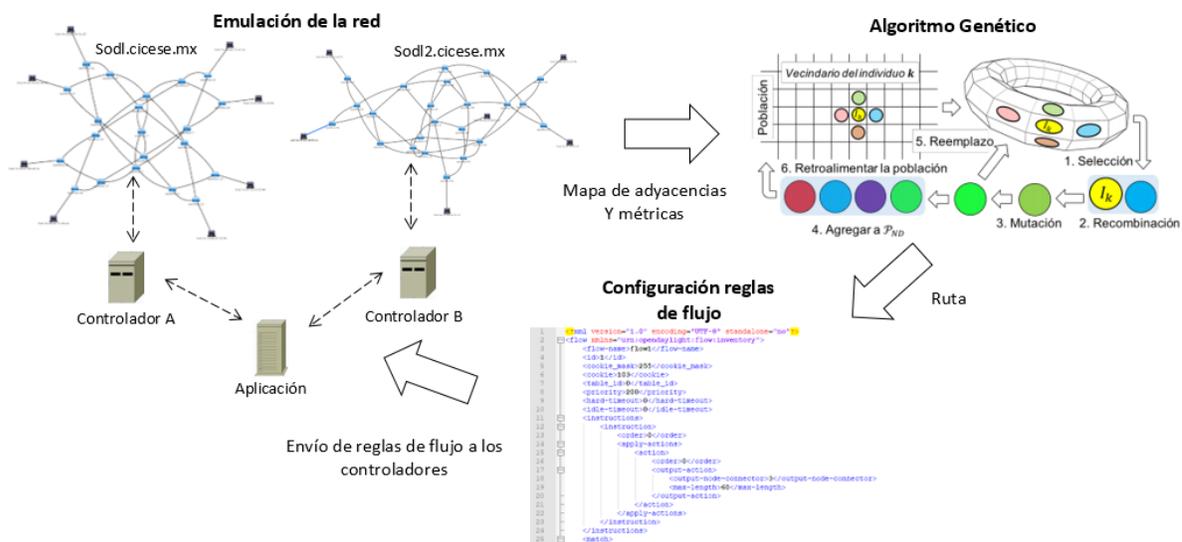
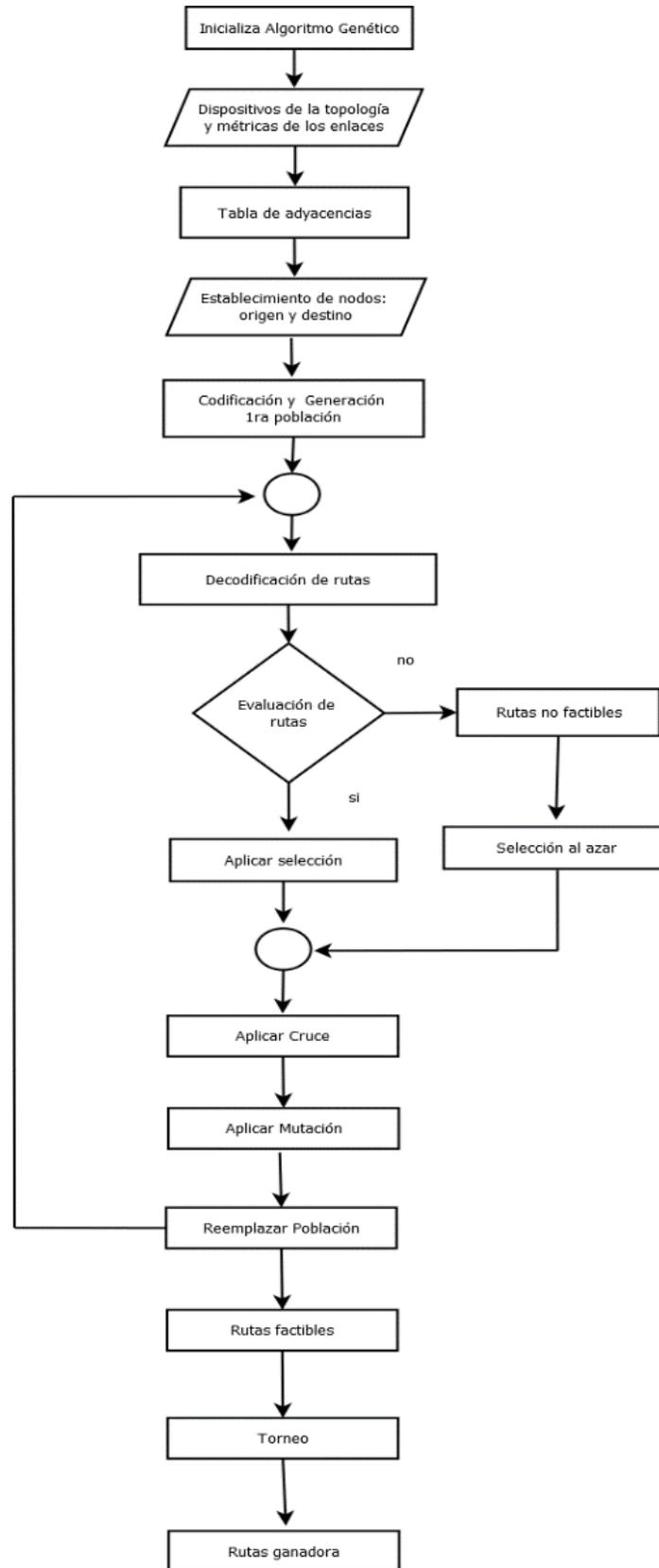


Figura 18. Procedimiento de enrutamiento propuesto.

El controlador entra en funcionamiento cuando un flujo llega a un interruptor (switch) y este no encuentra coincidencias en sus reglas de flujo. Entonces es enviado al controlador y es en este punto donde entra en funcionamiento nuestra propuesta de enrutamiento. Primero se determinan los nodos de origen y de destino, en nuestro caso los nodos en los extremos H1 y H14 de la figura 17. Como se

encuentran en dominios independientes se realiza la consulta en cada uno de los controladores y se abstraen los enlaces entre los diversos dispositivos como sus nombres mediante el protocolo NETCONF (anexo A), para realizar la tabla de adyacencia de la topología de la red. Estos datos son enviados al algoritmo genético, el cual se desarrolló en Matlab. El algoritmo se encarga tanto de calcular la mejor ruta de acuerdo con la función fitness como de realizar la tabla de adyacencias. En nuestra condición, la ruta capaz de transferir la mayor cantidad de información, contemplando las métricas de retardo y ancho de banda, sin olvidar el establecimiento del nivel de seguridad superior a 3.2.

Una vez que se obtiene una ruta favorable, se crean las reglas de flujo de cada dispositivo (Anexo B), configurando un número de puerto para cada flujo de acuerdo con el tipo de tráfico y un nivel de priorización. Finalmente, el flujo es dirigido a su destino por la ruta establecida por el algoritmo. La figura 19, muestra el diagrama a bloques del algoritmo genético implementado.



**Figura 19.** Diagrama a bloques del funcionamiento del algoritmo genético propuesto.

## Capítulo 5. Resultados y conclusiones

---

### 5.1 Introducción

En el capítulo 4 se describieron de manera detallada, el diagrama del algoritmo genético propuesto (figura 19), la topología implementada (figura 17) y el procedimiento de enrutamiento propuesto (figura 18). Teniendo como problema encontrar la mejor ruta para transferir la mayor cantidad de información entre los nodos de los extremos. En este capítulo se muestran los resultados de las rutas proporcionadas por el algoritmo genético propuesto, la evolución de la ruta proporcionada por el algoritmo y la predeterminada por el controlador.

### 5.2 Evaluación del algoritmo genético

Para establecer la ruta entre los extremos de la topología, es necesario realizar dos procesos de enrutamientos, uno por cada controlador. Es decir, que el enrutamiento se llevará a cabo en cada uno de los dominios independientemente. Una vez finalizados la aplicación de enrutamiento se encarga de unir las trayectorias y formar la ruta completa.

En cuanto a la configuración del algoritmo en ambos dominios se estableció una población de 50 individuos (posibles soluciones) y 25 interacciones; además de la función fitness con tres parámetros: mayor tasa de transferencia, menor retardo y una restricción en el nivel de seguridad  $S > 3.2$ .

Para el controlador A (dominio `sodl.cicese.mx`), se estableció el enrutamiento entre el nodo H1 y S1, mientras que para el controlador B (dominio `sodl2.cicese.mx`) entre los nodos S22 y H14. En la figura 20, se muestran las rutas encontradas en el controlador A y en las figuras 23 las rutas del controlador B. Las rutas representadas de color azul son las favorables con el nivel de seguridad y en rojo las que no cumplen con este criterio de selección.

El primer dominio se descubrieron 80 rutas de las cuales únicamente 32 fueron favorables con el nivel de seguridad establecido. Mientras que, en el segundo dominio fueron 315 rutas de las cuales 227 cumplieron con el nivel de seguridad.

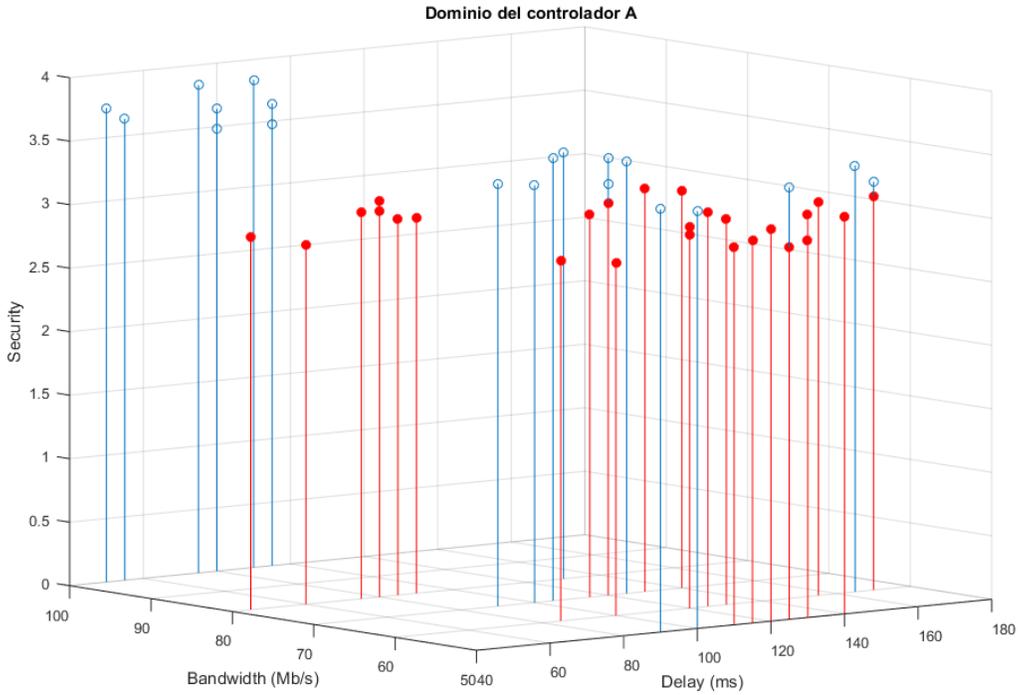


Figura 20. Rutas del controlador A. Las rutas azules son posibles soluciones favorables, mientras que las rojas son rutas que no cumplen con el nivel de seguridad requerido ( $s > 3.2$ ).

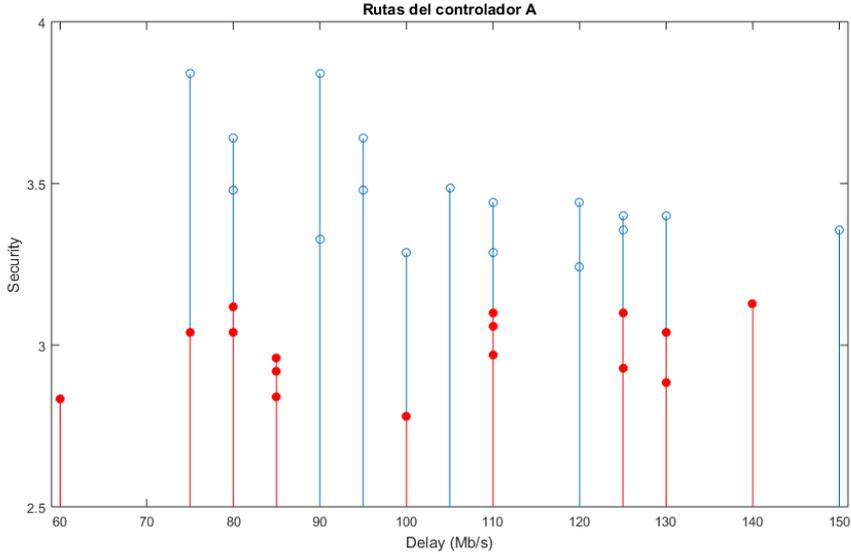
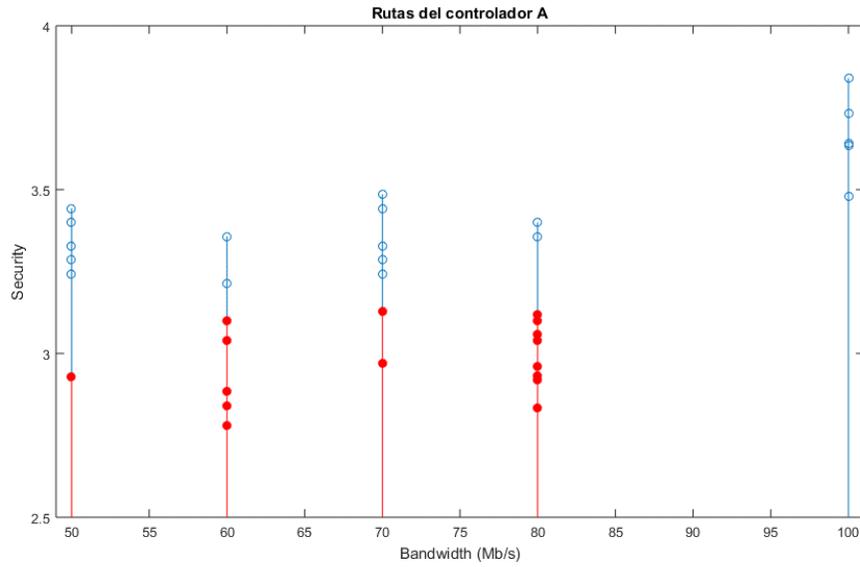
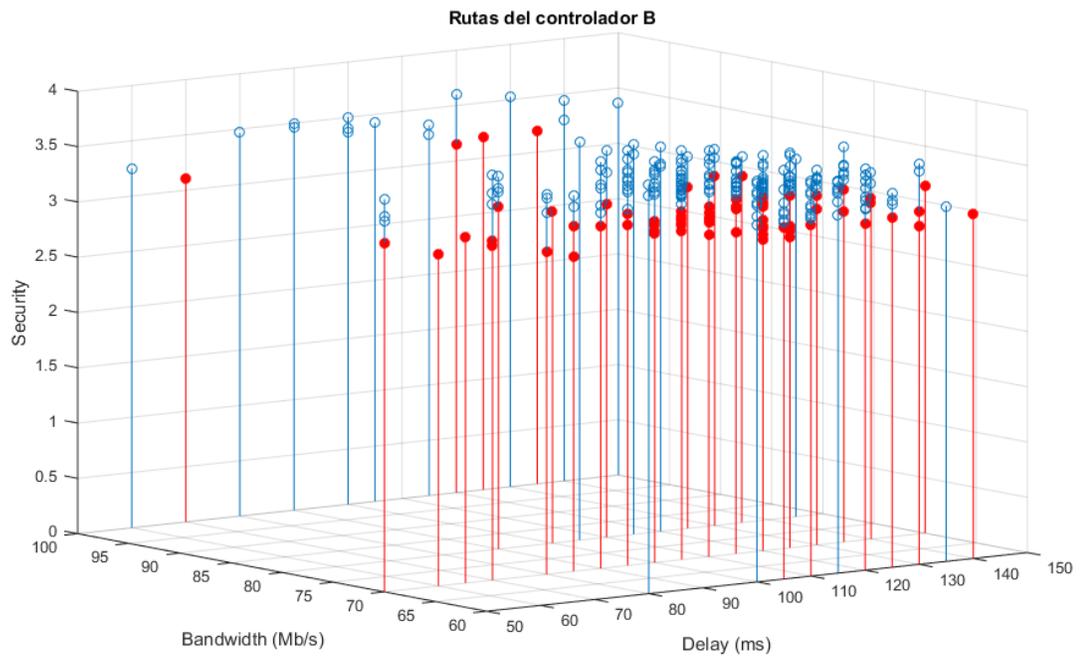


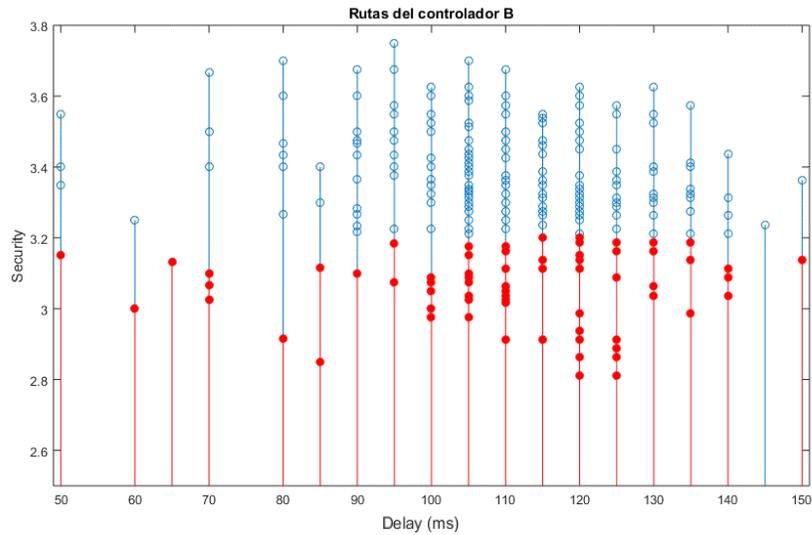
Figura 21. Rutas del controlador A. Delay (ms) / nivel de seguridad.



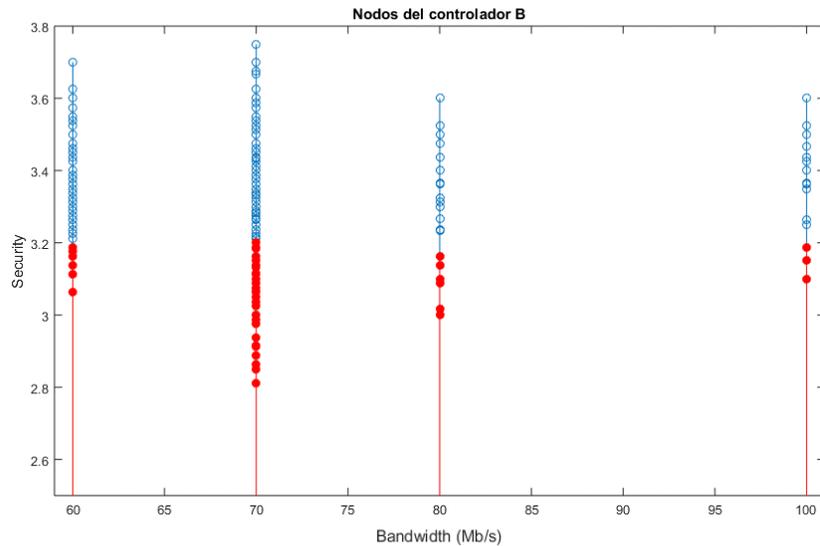
**Figura 22.** Rutas del controlador A. Bandwidth (Mb/s) /nivel de seguridad.



**Figura 23.** Rutas del controlador B. Las rutas azules son posibles soluciones favorables, mientras que las rojas son rutas que no cumplen con el nivel de seguridad requerido ( $s > 3.2$ ).



**Figura 24.** Rutas del controlador B. Delay (ms) / nivel de seguridad.



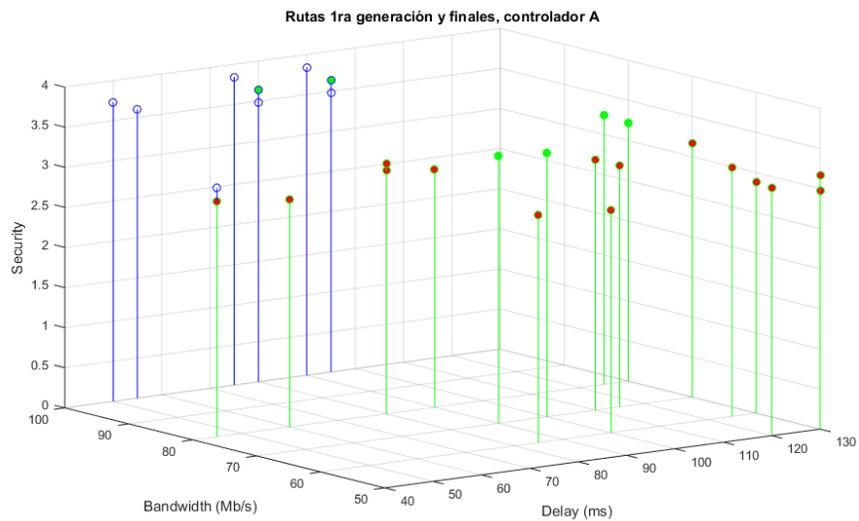
**Figura 25.** Rutas del controlador B. Bandwidth (Mb/s) / nivel de seguridad.

De las rutas competentes con el nivel de seguridad, únicamente son almacenadas las que cuentan con el mayor ancho de banda y menor retardo. Del controlador A se almacenaron 8 rutas y del controlador B 20. En la figura 26, se muestran las rutas del controlador A, en color verde están representadas las rutas pertenecientes a la primera generación y en azul las soluciones factibles obtenidas, mientras que en rojo están representadas las rutas cuyo nivel de confiabilidad no cumple con el requerimiento establecido. De esta misma manera se encuentran representadas las rutas del controlador B en la figura 27. Las métricas de las rutas factibles se encuentran en la tabla 4 y las de todas las rutas encontradas en el anexo c.

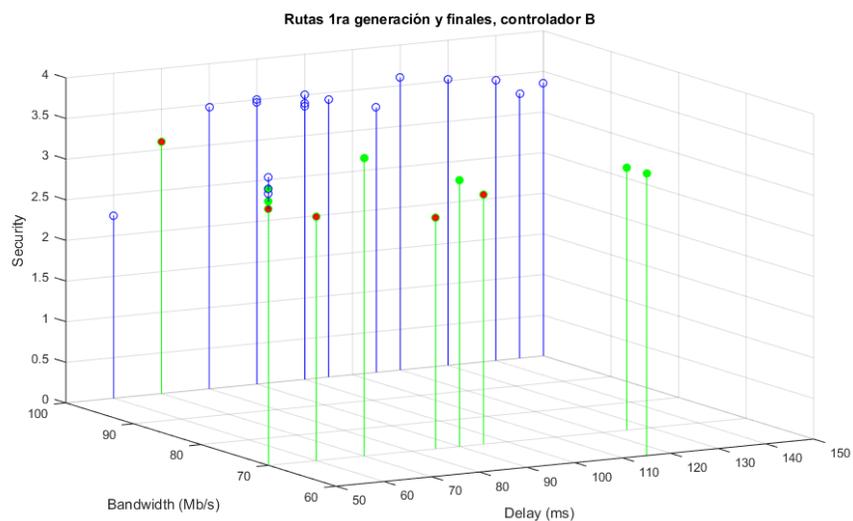
Una vez obtenidas las rutas favorables se realiza un proceso de torneo para determinar la mejor ruta para: transferir la mayor cantidad de información, contemplando tanto las métricas de retardo y el ancho de banda. Para ello se empleó la función minimizada del protocolo IGRP (ec. 21).

$$métrica = \frac{10^7}{BW} + \sum Delay \quad (21)$$

La mejor ruta en el controlador A es la número 5 y la número 17 para el controlador B de la tabla 4. Una vez obtenidas estas rutas, únicamente falta decodificar la trayectoria y configurar las reglas de flujo en el controlador.



**Figura 26.** Rutas del controlador A. En verde se muestran las rutas pertenecientes a la primera generación, en azul las rutas finales y en rojo las rutas que no cumplen con el nivel de seguridad.

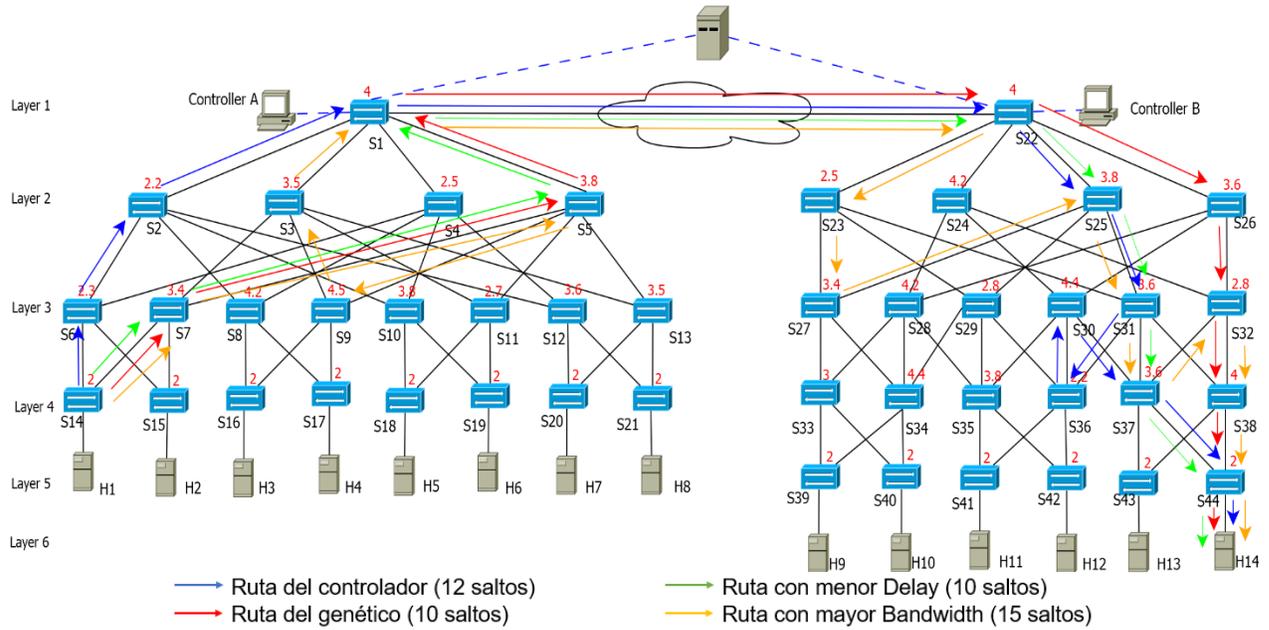


**Figura 27.** Rutas del controlador B. En verde se muestran las rutas pertenecientes a la primera generación, en azul las rutas finales y en rojo las rutas que no cumplen con el nivel de seguridad.

El algoritmo genético en este trabajo de tesis es capaz de elegir una ruta factible con criterios de confiabilidad, retardos y caudal eficaz. En nuestro caso específico, la ruta donde se pueda enviar la mayor cantidad de información considerando las siguientes métricas: nivel de seguridad, ancho de banda y retardos en la ruta (Delay y Jitter). En la figura 28, se muestran las rutas contempladas para evaluación: la ruta establecidas por el controlador, la ruta propuesta por el algoritmo genético, la ruta con mayor ancho de banda y la ruta con menor retardos.

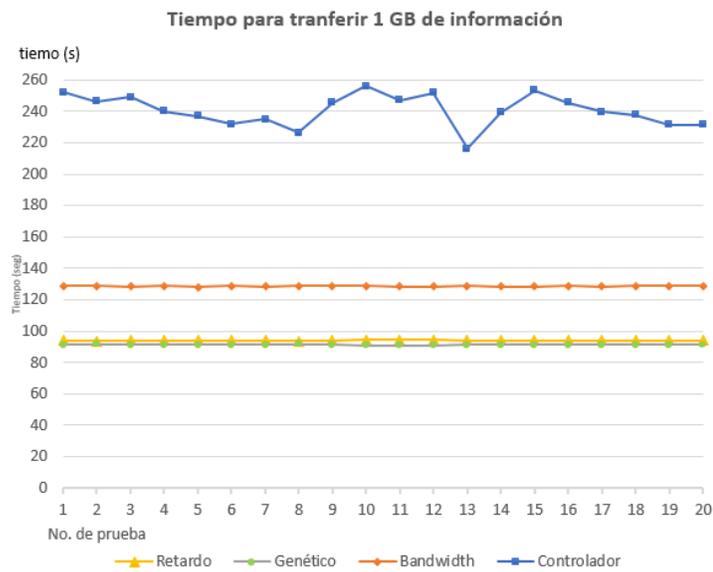
**Tabla. 4.** Rutas factibles de los controladores.

Ruta	Delay (ms)	Bandwidth (Mb/s)	Jitter (ms)	Security	Max Bandwidth/ Min Delay
Controlador A					
1	55	100	0	3.6	Min/Max
2	75	100	5	3.84	Max
3	80	100	0	3.48	Max
4	80	100	0	3.64	Max
5	50	100	0	3.73	Max
6	90	100	5	3.84	Max
7	95	100	0	3.48	Max
8	95	100	0	3.64	Max
Controlador B					
9	145	100	5	3.26	Max
10	115	100	5	3.26	Max
11	140	100	15	3.45	Max
12	150	100	10	3.36	Max
13	90	100	5	3.47	Max
14	100	100	10	3.5	Max
15	100	100	15	3.5	Max
16	50	70	10	3.55	Min
17	60	100	5	3.25	Max
18	105	100	5	3.42	Max
19	115	100	5	3.26	Max
20	120	100	15	3.6	Max
21	130	100	10	3.52	Max
22	100	100	5	3.36	Max
23	50	70	0	3.35	Min
24	120	100	15	3.6	Max
25	80	100	5	3.47	Max
26	90	100	10	3.5	Max
27	100	100	10	3.4	Max
28	50	70	5	3.4	Min



**Figura 28.** Rutas: controlador, algoritmo genético, mayor ancho de banda y menor retardo.

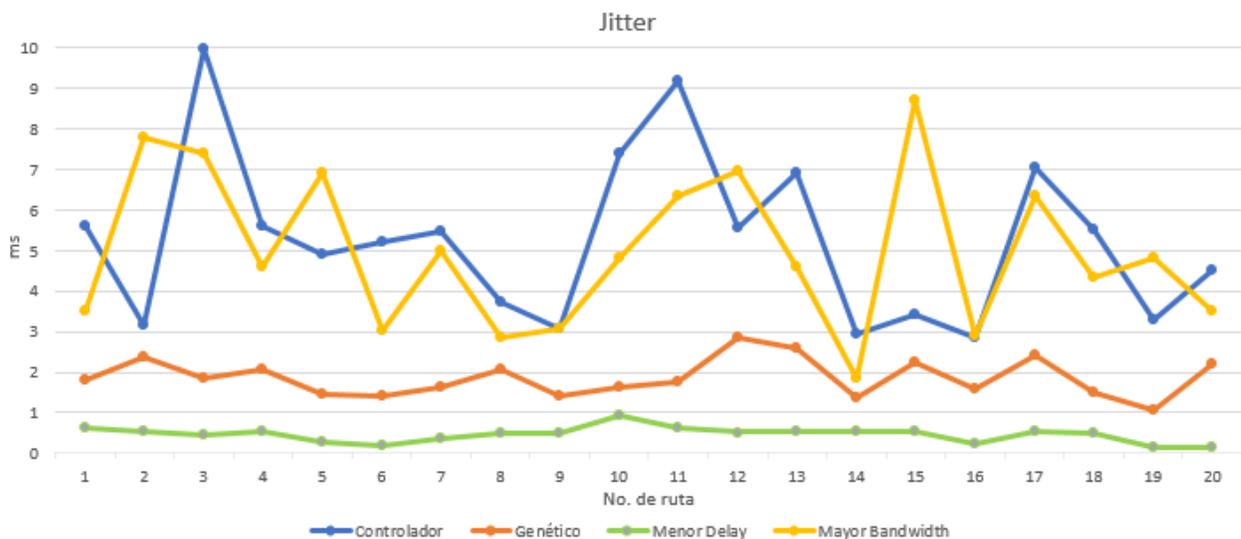
Para la evaluación de las rutas, se propone como primer experimente medir el tiempo en que cada ruta logra transferir 1 GB de información. La figura 29, muestra cada una de las evaluaciones; para esto se realizaron 20 pruebas, como se sugiere en la literatura y se utilizó la herramienta IPERF porque permite realizar mediciones con el máximo ancho de banda alcanzables por redes IP empleado los protocolos de comunicación TCP (protocolo de control de transmisión por sus siglas en inglés) y UDP (protocolo de datagramas de usuario por sus siglas en inglés).



**Figura 29.** Tiempo de estimación de cada ruta para transferir 1 GB de información.

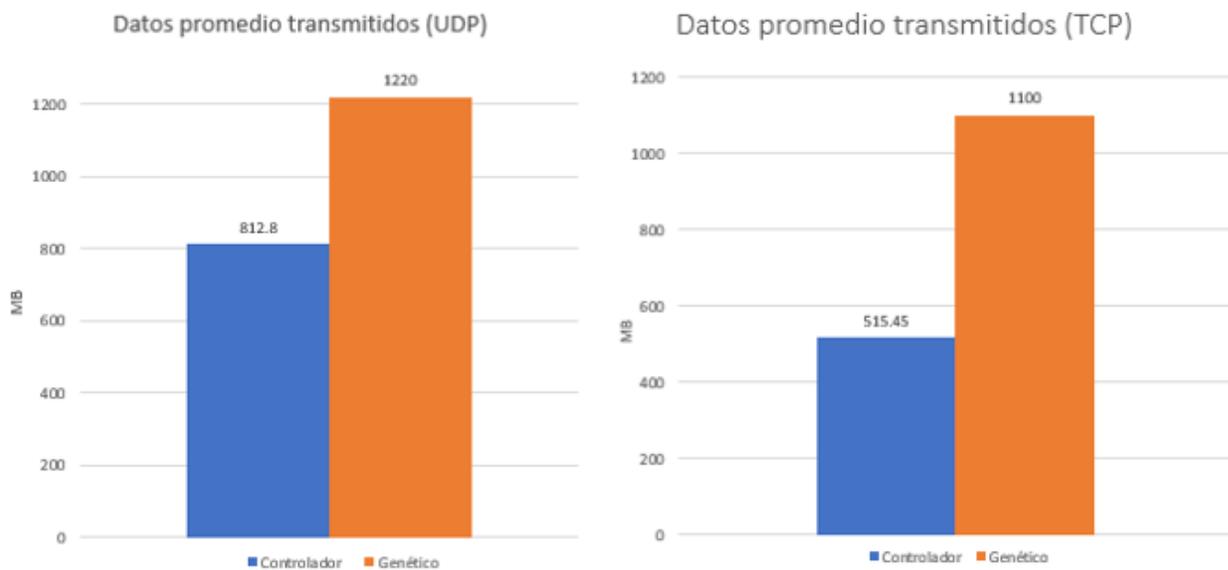
Como se puede observar la ruta con mayor demora es la establecida por el controlador, siendo la peor ruta para transferir información y con mayor número de fluctuaciones. Con una media de 240.31 segundos y una desviación estándar de 10.16 segundos. Mientras que la ruta propuesta por el algoritmo genético requiere menos de la mitad de tiempo para transferir la misma cantidad de información, con una media de 91.34 segundos y una desviación estándar de 0.20 segundos; seguida de las rutas que emplean las métricas de ancho de banda y retardo. La primera, con una media de 94 segundos y una desviación estándar de 0.20 segundos y la segunda con una media de 128.56 segundos y una desviación estándar de 0.22 segundos. Resultando ser mejor la ruta que considera múltiples métricas para el enrutamiento.

La aplicación IPERF, además, de ser una herramienta para medir el ancho de banda, también, permite ajustar y medir los parámetros relacionados con el tiempo en los protocolos TCP, UDP y SCTP con IPv4 e IPv6. Con esto, se realizaron las mediciones en las fluctuaciones durante el envío de datos (jitter) en cada una de las pruebas, como se muestra en la figura 30. La ruta con menos variaciones en el retardo es aquella que considera únicamente las métricas de retardo, con una media estándar de 0.45 segundos y una desviación estándar de 0.29 segundos. Mientras que las peores rutas, son las del controlador y el enrutamiento del ancho de banda, con una media de 5.08 y 4.97 segundos y una desviación estándar de 2.36 y 1.93 segundos. Teniendo un mejor desempeño la ruta propuesta por el algoritmo genético a estas dos rutas, con una media de 1.91 segundos y una desviación estándar de 0.56 segundos.



**Figura 30.** Jitter en cada una de las rutas.

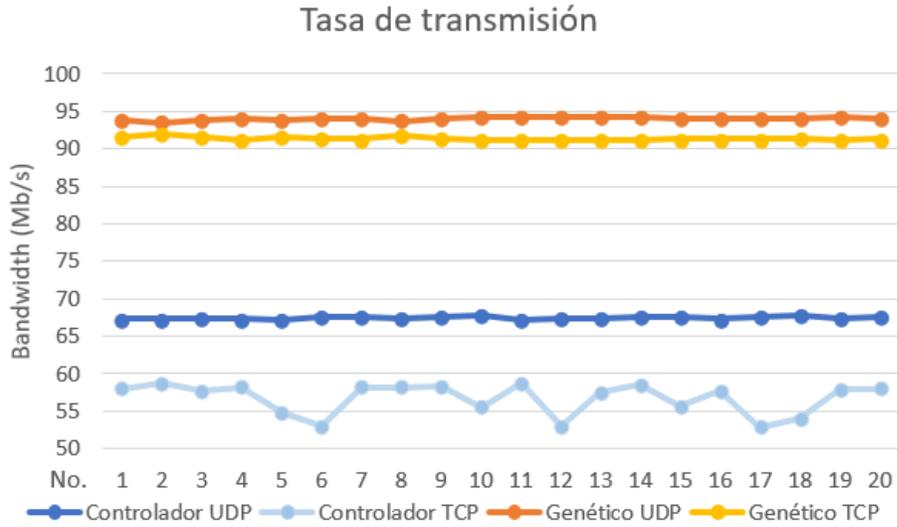
En la etapa final de experimentación, únicamente se seleccionaron las rutas del controlador y el algoritmo genético y observar las ventajas de utilizar una ruta orientada a la calidad de servicio con mejores métricas de ancho de banda, retardos y seguridad en comparación de una ruta proporcionada por un enrutamiento convencional del menor número de saltos, para ello se efectuó la mayor tasa de transferencia de información en las rutas con Iperf en un intervalo de 200 segundos tanto con el protocolos TCP como UDP. Como en los casos anteriores se realizaron 20 pruebas para cada protocolo; con UDP la ruta del controlador obtuvo una media de 812.8 MB y el algoritmo genético 1220 MB. Mientras con TCP el controlador redujo esta cantidad de información a 515.45 MB y el algoritmo genético a 1100 MB, como se muestra en la figura 31.



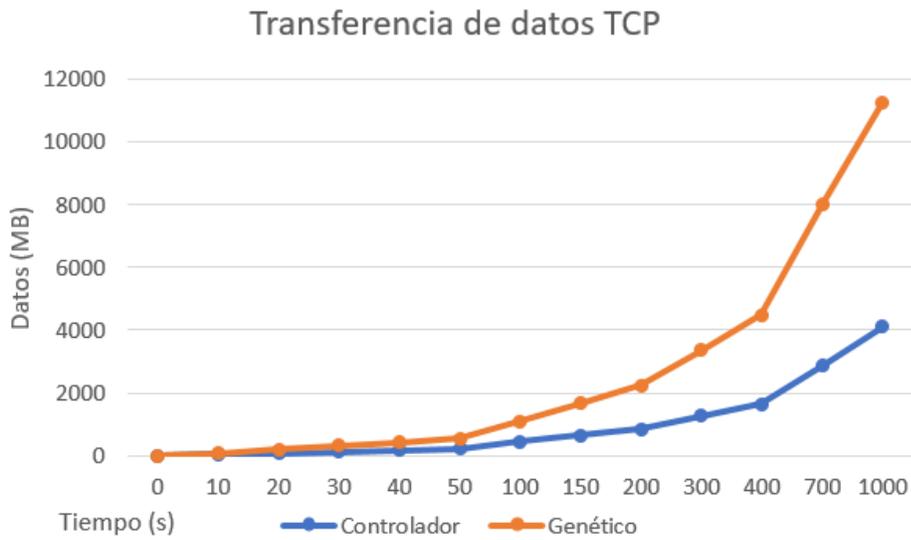
**Figura 31.** Datos promedio transferidos durante 200 segundos con los protocolos TCP y UDP.

La tasa de transferencia de datos o bandwidth (Mb/s), tanto del controlador y del algoritmo genético para cada prueba se muestra en la figura 32. Para el protocolo UDP la ruta del controlador tiene una media de 67.43 Mb/s y una desviación estándar de 0.14 segundos, la cual se reduce a una media de 56.75 Mb/s y una desviación estándar de 2.11 con el protocolo TCP. Mientras la ruta proporcionada por el algoritmo genético otorga una media de 94.04 Mb/s con tráfico UDP y 91.34 con TCP; en cuanto a sus desviaciones estándar ambas son de 0.20 segundos. La tasa de transferencia de datos se ve comprometida con el tipo de tráfico que se utilice en la ruta, siendo más notable esta diferencia en la ruta del controlador con una diferencia de 10.68 Mb/s, mientras con el algoritmo genético la diferencia es de tan solo 2.7 Mb/s.

En la figura 33, se ilustra la transferencia de datos con el protocolo TCP durante el periodo de prueba de 200 segundos y 1000 segundos obteniendo una transferencia de 4.11 GB con la ruta del controlador y 11.27 GB con el algoritmo genético.



**Figura 32.** Tasa de transmisión en cada una de las rutas durante 200 segundos.



**Figura 33.** Transferencia de datos TCP durante 1000 segundos.

### 5.3 Conclusiones

Los resultados de las pruebas realizadas demuestran que la selección de una trayectoria con múltiples métricas tiene un mayor rendimiento que las proporcionadas por el controlador y las rutas con el menor retardo y ancho de banda.

El algoritmo del controlador busca establecer una ruta sin importar las condiciones de la red, al no considerar los recursos disponibles de la red corre el riesgo de no proporcionar una adecuada ruta para un flujo y esté presente problemas de desempeño por la mala selección de recursos de la trayectoria o en su defecto la selección de una ruta congestionada. Este tipo de algoritmos únicamente se centran en seleccionar la ruta más corta por omisión, al igual que los métodos que utilizan una única métrica para el enrutamiento. Es por esto necesario la implementación de nuevos algoritmos que verifiquen el estado de la red antes de realizar el proceso de enrutamiento como se realiza en este trabajo.

El algoritmo genético propuesto proporciona un adecuado enrutamiento al tener una vista global de la red y abstraer las métricas QoS de la red. Las condiciones para la selección de la ruta son moderadas por los requerimientos solicitados por una aplicación. El algoritmo es adaptativo para cualquier condición de enrutamiento que se establezca, mientras el algoritmo del controlador no atiende ninguna de estas características esenciales para mejorar el servicio de enrutamiento.

El algoritmo genético brinda:

- Un aumento en el rendimiento de las aplicaciones de acuerdo con un enrutamiento adecuado.
- Un algoritmo adaptativo para QoS.
- Proporciona múltiples rutas que cumplen con las condiciones requeridas para los flujos.
- Escalabilidad, no tiene inconvenientes al agregar nuevos nodos a la topología.
- Funcionalidad en múltiples dominios.
- Rutas confiables.
- Disminuye retardos en el proceso de selección de rutas factibles empleando metaheurísticas.

## Capítulo 6. Resultados y conclusiones

---

### 6.1 Introducción

Durante los capítulos anteriores surgieron diversas cuestiones y problemas que se resolvieron mediante la investigación e implementación de diversas herramientas a lo largo de este trabajo como mininet y ODL en la emulación de la red SDN; Matlab en la implementación del algoritmo genético y proceso de enrutamiento multiobjetivo que se utilizaron en conjunto para la obtención de resultados. Siendo el objetivo principal de este capítulo, exponer tanto el análisis como las conclusiones de la investigación realizada, así como también las recomendaciones y las sugerencias del trabajo. La discusión se enfoca principalmente en la ruta propuesta por el algoritmo genético y la otorgada por el controlador. El proceso para elegir una trayectoria confiable y cumplir con ciertas condiciones de calidad de servicio. En nuestro caso, nos centramos en encontrar la mejor trayectoria para transmitir información, brindando así, una solución sólida al planteamiento del problema establecido con la factibilidad de llevar a cabo la réplica del procedimiento y obtención de los resultados presentados. Finalmente se propone el trabajo a futuro, tomando como punto de partida la investigación realizada para la aplicación de distintos escenarios de enrutamiento y aplicaciones en todo tipo de redes.

### 6.2 Análisis e implementación del algoritmo genético

Como se ha mostrado en el capítulo 4 y 5, la aplicación de un algoritmo genético para el proceso de enrutamiento orienta a la calidad de servicio es congruente al tratarse de un problema multiobjetivo, este tipo de algoritmos son buenos para proporcionar soluciones factibles en un tiempo relativamente corto. El algoritmo evolutivo consigue su meta de acuerdo con su función fitness, descarta las soluciones que no cumplan con el nivel de seguridad establecido y selecciona las rutas con menor retardo y mayor ancho de banda para posteriormente elegir la mejor para transmitir información.

De acuerdo con la metaheurística de población del algoritmo genético se obtienen mejores soluciones con un mayor número de interacciones, estableciendo la población con 50 individuos y 20 interacciones. En la primera población del controlador A, de los 50 individuos únicamente 21 fueron rutas factibles y solo 6 rutas cumplían con el nivel de seguridad; sin embargo, ninguna de ellas resultó ser la mejor ruta, como se

muestra en la figura 26. Por ello es favorable el proceso característico de los algoritmos evolutivos y genéticos (selección, cruzamiento y mutación), incrementando cuatro veces más el número de soluciones factibles y encontrando 8 soluciones óptimas. En cuanto al controlador B, en la primera generación 11 rutas fueron factibles y solo 4 de ellas cumplieron con el nivel de seguridad, pero lejos de pertenecer a las mejores rutas (figura 27). En este caso se incrementaron las soluciones 30 veces más y se encontraron 20 soluciones óptimas. Con esto podemos demostrar que el algoritmo genético puede encontrar mejores soluciones al problema a partir de sus procesos de cruzamiento y mutación a partir de una población de rutas aleatoriamente.

### **6.3 Análisis del algoritmo de enrutamiento en SDN**

La evaluación del comportamiento de las rutas se encuentra definida por las diversas condiciones de las trayectorias que permiten comprender el impacto que provoca el uso de métricas de calidad de servicio en el proceso de enrutamiento, así como la mejora que se puede obtener en el desempeño de la red y el rendimiento de las aplicaciones en una red SDN.

Para los casos de evaluación presentados en el capítulo 5. Se evalúa el rendimiento en el proceso de enrutamiento por el controlador y las métricas de retardo y ancho de banda, independientes y en conjunto como lo realiza nuestro algoritmo propuesto. En el caso de transferir la mayor cantidad de información de extremo a extremo en la red propuesta (figura 17), las rutas que consideran las métricas de QoS obtuvieron un mejor desempeño que la ruta del controlador. Mientras con el algoritmo genético se transmitió más del doble de información con respecto al controlador; con el protocolo TCP se tiene una ganancia del 52.15% y 33.38% con UDP.

En cuanto a la evaluación de tiempo para transmitir un GB de información. La ruta menos eficiente es de nuevo la proporcionada por el controlador con un tiempo promedio de 240.31 segundos, seguida por la ruta con el mayor ancho de banda reduciendo el tiempo en 111.75 segundos; mientras la ruta con el menor retardo en 146.31 segundos y la ruta del algoritmo genético en 148.97 segundos, siendo 2.6 veces menor el tiempo de transferencia al del controlador.

La consideración de emplear más de una métrica en el proceso de enrutamiento es un acierto, mejora el rendimiento de la red y reduce la probabilidad de degradación de un servicio, sin embargo, solo debe ser empleada cuando sea necesario para una gestión adecuada de los recursos y servicio.

## **6.4 Aportaciones**

La aportación principalmente de este trabajo de investigación es la adaptación de un algoritmo genético (mocell), para la selección de trayectorias confiables con base a distintas métricas de QoS de la red. La selección de este algoritmo se realizó para simplificar el problema de complejidad multivariable y obtener rutas que satisfagan los requerimientos QoS establecidos por aplicaciones como, retardos, latencia y ancho de banda. Además, de proporcionar un nivel de confiabilidad para las rutas, proceso que ningún algoritmo de enrutamiento convencional realiza. Al emplear un algoritmo genético se reduce el proceso de selección de rutas factibles con múltiples parámetros empleando metaheurísticas y métodos de selección basados en máximos y mínimos evitando emplear procesos que requieren de una demanda grande de recursos y tiempo de procesamiento.

## **6.5 Trabajo futuro**

La implementación de QoS en SDN es de gran prioridad sobre todo en el enrutamiento, por lo cual como trabajo a futuro se recomienda implementación de una herramienta o módulo que permita la abstracción de los requerimientos de las aplicaciones para realizar el proceso de adaptación en la función fitness del algoritmo de enrutamiento. Además de emplear una herramienta de monitoreo constante de las métricas de QoS para mejorar la precisión en el cálculo de las trayectorias y evitar el congestionamiento de la red.

También es necesario mejorar e implementar de manera automatizada el proceso de abstracción del estado de la red y escritura de las reglas de flujo en el controlador ya que el proceso de enrutamiento se ve afectado por estas dos etapas que se retroalimentan de manera manual.

Se recomienda realizar la evolución del algoritmo con diferentes niveles de tráfico y protocolos de comunicación, así como la consideración de ampliar el vector objetivo con otras métricas en el proceso de

desempeño como la pérdida de datos, además de emplear otras metaheurísticas aparte de la de población y modificar la estructura de toma de decisiones y el proceso de selección de rutas factibles.

Durante el desarrollo del trabajo también se observó que hacen falta la implementación de métricas relacionadas con la seguridad de los nodos y un sistema de evaluación estandarizado para la confiabilidad de las redes, siendo una línea de trabajo con mucho futuro al igual que la implementación de calidad de servicio multiobjetivo en SDN. Son pocas las propuestas en este ámbito de estudio y aun no son explotadas ampliamente.

## Literatura citada

---

- Abe, J. O., Mantar, H. A., Yayimli, A. G. 2015. K-Maximally Disjoint Path Routing Algorithms for SDN. Proceedings - 2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC 2015, 499–508. doi:10.1109/CyberC.2015.45
- Agarwal, K., Rozner, E., Dixon, C., Carter, J. 2014. SDN traceroute. 145–150. doi:10.1145/2620728.2620756
- Ahmed, H. G., Ramalakshmi, R. 2018. Distributed SDN Controllers for Load Balancing Application. 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI), (Icoei), 758–764.
- Akyildiz, I. F., Lee, A., Wang, P., Luo, M., Chou, W. 2014. A roadmap for traffic engineering in software defined networks. Computer Networks, 71, 1–30. doi:10.1016/j.comnet.2014.06.002
- Bannour, F., Souihi, S., Mellouk, A. 2018. Distributed SDN Control: Survey, Taxonomy, and Challenges. IEEE Communications Surveys and Tutorials, 20(1), 333–354. doi:10.1109/COMST.2017.2782482
- Basit, A., Ahmed, N. 2017. Path diversity for Inter-domain Routing security. Proceedings of 2017 14th International Bhurban Conference on Applied Sciences and Technology, IBCAST 2017, 384–391. doi:10.1109/IBCAST.2017.7868083
- Bays, L. R., Oliveira, R. R., Barcellos, M. P., Gaspar, L. P., Mauro Madeira, E. R. 2015. Virtual network security: threats, countermeasures, and challenges. Journal of Internet Services and Applications, 6(1), 1–19. doi:10.1186/s13174-014-0015-z
- Braden, R., Clark, D., Shenker, S. 1994. RFC-1633: Integrated Services in the Internet Architecture: an Overview Status of this Memo. Internet Research, 1–28.
- Casado, M., Freedman, M. J., Pettit, J., Luo, J., McKeown, N., Shenker, S. 2007. Ethane: Taking control of the enterprise. Computer Communication Review, 37(4), 1–12. doi:10.1145/1282427.1282382
- Coello, C. A. C., Lamont, G. B., Veldhuizen, D. A. Van. 2007. Evolutionary Algorithms for Solving Multi-Objective Problems. En Evolutionary Algorithms for Solving Multi-Objective Problems. doi:10.1007/978-0-387-36797-2
- De Tre, G., Hallez, A., Bronselaer, A. 2014. Performance optimization of object comparison. International Journal of intelligent Systems, 29(2), 495–524. doi:10.1002/int
- Djouani, R., Djouani, K., Boutekkouk, F., Sahbi, R. 2019. A Security Proposal for IoT integrated with SDN and Cloud. Proceedings - 2018 International Conference on Wireless Networks and Mobile Communications, WINCOM 2018, 1–5. doi:10.1109/WINCOM.2018.8629727
- Doshi, M., Kamdar, A., Kansara, K. 2018. Multi-constraint qos disjoint multipath routing in SDN. Advances in Intelligent Systems and Computing, 810, 377–387. doi:10.1007/978-981-13-1513-8\_40
- Egilmez, H. E., Dane, S. T., Bagci, K. T., Tekalp, A. M. 2012. OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end Quality of Service over Software-Defined Networks. 2012 Conference Handbook - Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, APSIPA ASC 2012, 1–8.

- Eiben, A. E., Smith, J. E. 2015. Natural Computing Series Introduction to Evolutionary Computing. En Natural Computing Series. doi:10.1007/978-3-662-44874-8
- Firoiu, V., Le Boudec, J. Y., Towsley, D., Zhang, Z. L. I. 2002. Theories and models for internet quality of service. *Proceedings of the IEEE*, 90(9), 1565–1591. doi:10.1109/JPROC.2002.802002
- F. W. K., -, T. J. 2010. A Survey on QoS in Next Generation Networks. *INTERNATIONAL JOURNAL ON Advances in Information Sciences and Service Sciences*, 2(4), 91–102. doi:10.4156/aiss.vol2.issue4.10
- Greenberg, A., Hjalmtysson, G., Maltz, D. A., Myers, A., Rexford, J., Xie, G., Yan, H., Zhan, J., Zhang, H. 2005. Public Review for A Clean Slate 4D Approach to Network Control and Management *a c m s i g c o m m*. 35(5), 41–54.
- Haleplidis, E., Hadi Salim, J., Denazis, S., Koufopavlou, O. 2015. Towards a Network Abstraction Model for SDN. *Journal of Network and Systems Management*, 23(2), 309–327. doi:10.1007/s10922-014-9319-3
- Henneke, D., Wisniewski, L., Jasperneite, J. 2016. Analysis of realizing a future industrial network by means of Software-Defined Networking (SDN). *IEEE International Workshop on Factory Communication Systems - Proceedings, WFCS, 2016-June*. doi:10.1109/WFCS.2016.7496525
- Hoang, D. B., Pham, M. 2015. On software-defined networking and the design of SDN controllers. *2015 International Conference on the Network of the Future, NOF 2015*, 1–3. doi:10.1109/NOF.2015.7333307
- Hong, W., Wang, K., Hsu, Y. H. 2013. Application-aware resource allocation for SDN-based cloud datacenters. *Proceedings - 2013 International Conference on Cloud Computing and Big Data, CLOUDCOM-ASIA 2013*, 106–110. doi:10.1109/CLOUDCOM-ASIA.2013.44
- Jarraya, Y., Madi, T., Debbabi, M. 2014. A survey and a layered taxonomy of software-defined networking. *IEEE Communications Surveys and Tutorials*, 16(4), 1955–1980. doi:10.1109/COMST.2014.2320094
- Kreutz, D., Ramos, F. M. V., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., Uhlig, S. 2015. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1), 14–76. doi:10.1109/JPROC.2014.2371999
- Lamali, M. L., Fergani, N., Cohen, J., Pouyllau, H. 2016. Path computation in multi-layer networks: Complexity and algorithms. *Proceedings - IEEE INFOCOM, 2016-July*, 1–9. doi:10.1109/INFOCOM.2016.7524550
- Li, K., Ou, Q., Chen, H., Zou, B. 2018. Controller Cluster-Based Interconnecting for Multi-domain SDN Networks. *Proceedings - 2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC 2017, 2018-Janua*, 482–485. doi:10.1109/CyberC.2017.41
- Li, X., Wu, H., Gruenbacher, D., Scoglio, C., Anjali, T. 2016. Efficient routing for middlebox policy enforcement in software-defined networking. *Computer Networks*, 110, 243–252. doi:10.1016/j.comnet.2016.10.002
- Mao, J., Liu, J., Qi, C., Wang, M., Cheng, H., Chen, J. 2017. RouteGuardian: Constructing secure routing paths in software-defined networking. *Tsinghua Science and Technology*, 22(4), 400–412. doi:10.23919/tst.2017.7986943

- Narisetty, R., Dane, L., Malishevskiy, A., Gurkan, D., Bailey, S., Narayan, S., Mysore, S. 2013. OpenFlow configuration protocol: Implementation for the of management plane. Proceedings - 2013 2nd GENI Research and Educational Experiment Workshop, GREE 2013, (March), 66–67. doi:10.1109/GREE.2013.21
- Nguyen, K., Yamada, S. 2016. An experimental feasibility study on applying SDN technology to disaster-resilient wide area networks. *Annales des Telecommunications/Annals of Telecommunications*, 71(11–12), 639–647. doi:10.1007/s12243-016-0502-2
- Paliwal, M., Shrimankar, D., Tembhurne, O. 2018. Controllers in SDN: A review report. *IEEE Access*, 6, 36256–36270. doi:10.1109/ACCESS.2018.2846236
- Papadimitratos, P., Haas, Z. J. 2002. Securing the internet routing infrastructure. *IEEE Communications Magazine*, 40(10), 60–68. doi:10.1109/MCOM.2002.1039858
- Park, T., Kim, Y., Park, J., Suh, H., Hong, B., Shin, S. 2016. QoSE: Quality of security a network security framework with distributed NFV. 2016 IEEE International Conference on Communications, ICC 2016, 1–6. doi:10.1109/ICC.2016.7510777
- Patricia, D., Levy, B. 2011. Centro de investigación científica y de educación superior de ensenada, baja california. 01(3918), 22860.
- Pattaranantakul, M., He, R., Meddahi, A., Zhang, Z. 2016. SecMANO: Towards network functions virtualization (NFV) based security MANagement and orchestration. Proceedings - 15th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, 10th IEEE International Conference on Big Data Science and Engineering and 14th IEEE International Symposium on Parallel and Distributed Processing with Applications, IEEE TrustCom/BigDataSE/ISPA 2016, 598–605. doi:10.1109/TrustCom.2016.0115
- Raayatpanah, M. A., Salehi Fathabadi, H., Khalaj, B. H., Khodayifar, S., Pardalos, P. M. 2014. Bounds on end-to-end statistical delay and jitter in multiple multicast coded packet networks. *Journal of Network and Computer Applications*, 41(1), 217–227. doi:10.1016/j.jnca.2013.12.004
- Ramos, F. M. V., Kreutz, D., Veríssimo, P. 2015. Software-defined networks: On the road to the softwarization of networking. *Cutter IT Journal*, 28(5), 6–13.
- Zhang, H., Yan, J. 2015. Performance of SDN Routing in Comparison with Legacy Routing Protocols. Proceedings - 2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC 2015, 491–494. doi:10.1109/CyberC.2015.30

## Anexos A

Tabla. 5. Abstracción de nodos en mininet.

<b>Controlador A</b>			
<b>Nodo en la red</b>	<b>Nombre</b>	<b>Nodo en la red</b>	<b>Nombre</b>
{'host:0e:db:f6:2c:21:5a"	H8	"openflow:21"	S15
"host:12:f3:48:26:92:33"	H7	"openflow:22"	S16
"host:1e:70:ba:b1:2c:ee"	H6	"openflow:23"	S17
"host:26:16:5f:e5:cd:e9"	H5	"openflow:24"	S18
"host:56:68:70:63:f1:97"	H4	"openflow:25"	S19
"host:66:47:eb:6b:45:fb"	H3	"openflow:3"	S3
"host:9a:73:b9:7e:73:c9"	H2	"openflow:32"	S20
"host:e6:38:d0:24:c2:13"	H1	"openflow:33"	S21
"openflow:1"	S1	"openflow:34"	S14
"openflow:16"	S10	"openflow:4"	S4
"openflow:17"	S11	"openflow:5"	S5
"openflow:18"	S12	"openflow:6"	S6
"openflow:19"	S13	"openflow:7"	S7
"openflow:2"	S2	"openflow:8"	S8
"openflow:20"	S14	"openflow:9"	S9
<b>Controlador B</b>			
"host:16:64:90:c9:7f:03"	H7	"openflow:23"	S38
"host:26:f2:13:e9:23:b8"	H12	"openflow:24"	S39
"host:46:61:93:a7:f9:c2"	H9	"openflow:25"	S40
"host:a6:4c:ec:45:57:87"	H13	"openflow:3"	S24
"host:c2:c5:a5:35:e6:40"	H10	"openflow:32"	S41
"host:f2:1a:f1:11:e9:a8"	H11	"openflow:33"	S42
"openflow:1"	S22	"openflow:34"	S43
"openflow:16"	S31	"openflow:35"	S44
"openflow:17"	S32	"openflow:4"	S25
"openflow:18"	S33	"openflow:5"	S26
"openflow:19"	S34	"openflow:6"	S27
"openflow:2"	S23	"openflow:7"	S28
"openflow:20"	S35	"openflow:8"	S29
"openflow:21"	S36	"openflow:9"	S30
"openflow:22"	S37		

## Anexos B

---

Reglas de flujo para ruta (algoritmo genético).

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

```
#para configurar un flujo a traves de ODL en un switch
```

```
# #controlador ODL Dominio A (sodl)
```

```
curl -X PUT -d @h1 -H "Content-Type: application/xml" -H "Accept: application/xml" --user admin:admin
http://sodl.cicese.mx:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:20/table/0/flow/1
curl -X PUT -d @h2 -H "Content-Type: application/xml" -H "Accept: application/xml" --user admin:admin
http://sodl.cicese.mx:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:7/table/0/flow/1
curl -X PUT -d @h3 -H "Content-Type: application/xml" -H "Accept: application/xml" --user admin:admin
http://sodl.cicese.mx:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:3/table/0/flow/1
curl -X PUT -d @h4 -H "Content-Type: application/xml" -H "Accept: application/xml" --user admin:admin
http://sodl.cicese.mx:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:19/table/0/flow/1
curl -X PUT -d @h5 -H "Content-Type: application/xml" -H "Accept: application/xml" --user admin:admin
http://sodl.cicese.mx:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:5/table/0/flow/1
curl -X PUT -d @h6 -H "Content-Type: application/xml" -H "Accept: application/xml" --user admin:admin
http://sodl.cicese.mx:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:1/table/0/flow/1
```

```
# #controlador ODL Dominio A (sodl)
```

```
curl -X PUT -d @h7 -H "Content-Type: application/xml" -H "Accept: application/xml" --user admin:admin
http://sodl2.cicese.mx:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:34/table/0/flow/1
curl -X PUT -d @h8 -H "Content-Type: application/xml" -H "Accept: application/xml" --user admin:admin
http://sodl2.cicese.mx:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:35/table/0/flow/1
curl -X PUT -d @h9 -H "Content-Type: application/xml" -H "Accept: application/xml" --user admin:admin
http://sodl2.cicese.mx:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:39/table/0/flow/1
curl -X PUT -d @h10 -H "Content-Type: application/xml" -H "Accept: application/xml" --user admin:admin
http://sodl2.cicese.mx:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:37/table/0/flow/1
curl -X PUT -d @h11 -H "Content-Type: application/xml" -H "Accept: application/xml" --user admin:admin
http://sodl2.cicese.mx:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:49/table/0/flow/1
curl -X PUT -d @h12 -H "Content-Type: application/xml" -H "Accept: application/xml" --user admin:admin
http://sodl2.cicese.mx:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:55/table/0/flow/1
curl -X PUT -d @h13 -H "Content-Type: application/xml" -H "Accept: application/xml" --user admin:admin
http://sodl2.cicese.mx:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:50/table/0/flow/1
curl -X PUT -d @h14 -H "Content-Type: application/xml" -H "Accept: application/xml" --user admin:admin
http://sodl2.cicese.mx:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:56/table/0/flow/1
curl -X PUT -d @h15 -H "Content-Type: application/xml" -H "Accept: application/xml" --user admin:admin
http://sodl2.cicese.mx:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:68/table/0/flow/1
```

Formato XML. Es necesario cambiar el número de puerto de acuerdo con la dirección del flujo.

```
<?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>3</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>
```

## Anexo C

---

**Tabla. 6.** Métricas de las rutas encontradas por el controlador A.

Delay (ms)	Bandwidth (Mb/s)	Jitter (ms)	Security	Delay (ms)	Bandwidth (Mb/s)	Jitter (ms)	Security
60	2.833333333	5	80	55	3.633333333	0	100
75	3.04	5	80	75	3.84	5	100
120	3.242857143	10	50	125	3.4	5	50
100	3.285714286	5	50	110	3.442857143	5	50
120	3.242857143	10	70	125	3.4	5	70
100	3.285714286	5	70	110	3.442857143	5	70
90	2.96	5	80	80	3.48	0	100
125	2.928571429	5	70	120	3.085714286	0	70
130	2.971428571	5	70	110	3.128571429	0	70
125	2.928571429	5	50	120	3.085714286	0	50
130	2.971428571	5	50	110	3.128571429	0	50
80	3.12	5	80	80	3.64	0	100
125	3.014285714	5	80	130	3.171428571	0	80
110	3.057142857	5	80	130	3.357142857	0	80
120	3.014285714	5	80	125	3.171428571	0	80
105	3.057142857	5	80	125	3.357142857	0	80
45	2.933333333	5	80	50	3.733333333	0	100
80	3.04	5	80	90	3.84	5	100
110	3.285714286	10	50	120	3.442857143	0	50
90	3.328571429	5	50	105	3.485714286	0	50
110	3.285714286	10	70	120	3.442857143	0	70
90	3.328571429	5	70	105	3.485714286	0	70
85	2.96	5	80	95	3.48	0	100
110	2.971428571	5	70	140	3.128571429	0	70
115	3.014285714	5	70	130	3.171428571	0	70
110	2.971428571	5	50	140	3.128571429	0	50
115	3.014285714	5	50	130	3.171428571	0	50
85	2.92	5	80	95	3.64	0	100
130	3.057142857	5	80	130	3.214285714	0	80
115	3.1	5	80	130	3.4	0	80
125	3.057142857	5	80	125	3.214285714	0	80
110	3.1	5	80	125	3.4	0	80
130	3.04	5	60	100	2.78	0	60
150	3.357142857	10	60	115	2.942857143	0	60

155	3.1	5	60	130	2.885714286	0	60
155	3.214285714	5	60	120	3	0	60
125	3.1	5	60	85	2.84	0	60
165	3.357142857	10	60	120	2.942857143	0	60
170	3.1	5	60	125	2.885714286	0	60
170	3.214285714	5	60	125	2.857142857	0	60

**Tabla. 7.** Métricas de las rutas encontradas por el controlador B.

Delay (ms)	Bandwidth (Mb/s)	Jitter (ms)	Security	Delay (ms)	Bandwidth (Mb/s)	Jitter (ms)	Security
120	3.2875	10	70	105	3.3	0	70
105	3.0875	15	70	115	3.35	5	100
110	3.0375	10	70	105	3.525	15	60
120	3.1875	10	70	95	3.75	10	70
110	3.3625	5	70	95	3.55	0	70
140	3.2625	5	100	105	3.6	5	80
130	3.1625	5	80	95	3.675	5	70
90	3.216666667	0	70	105	3.475	0	70
105	3.4125	10	70	115	3.525	5	80
105	3.2125	0	70	105	3.7	15	60
115	3.2625	5	100	120	3.3125	0	70
105	3.3375	5	70	115	3.525	5	70
115	3.1375	0	70	100	3.325	10	70
125	3.1875	5	100	105	3.275	5	70
115	3.3625	15	60	115	3.1125	15	70
105	3.5875	10	70	105	3.2875	10	70
105	3.3875	0	70	135	3.1875	10	100
115	3.4375	5	80	125	3.0875	10	80
105	3.5125	5	70	85	3.116666667	5	70
115	3.3125	0	70	115	3.3125	5	70
125	3.3625	5	80	90	3.675	15	70
115	3.5375	15	60	90	3.475	5	70
115	3.3125	0	70	100	3.525	10	70
125	3.3625	5	70	90	3.6	10	70
110	3.1625	10	70	100	3.4	5	70
115	3.1125	5	70	110	3.45	10	70
115	3.1125	10	70	100	3.625	20	60
135	3.1875	5	100	100	3.6	25	60
125	3.0875	5	80	100	3.4	15	60

85	3.116666667	0	70	110	3.45	20	60
120	3.3125	0	70	100	3.525	20	60
105	3.5125	10	70	110	3.325	15	60
105	3.3125	0	70	120	3.375	20	60
115	3.3625	5	70	110	3.425	10	70
105	3.4375	5	70	120	3.475	15	80
115	3.2375	0	70	120	3.35	5	70
125	3.2875	5	70	130	3.4	10	80
115	3.4625	15	60	130	3.4	15	80
115	3.4375	20	60	120	3.3	15	80
115	3.2375	10	60	80	3.4	10	70
125	3.2875	15	60	100	3.075	10	70
115	3.3625	15	60	105	3.025	5	70
125	3.1625	10	60	105	3.075	10	70
135	3.2125	15	60	110	3.025	5	70
125	3.2625	5	70	95	3.225	15	70
135	3.3125	10	80	105	3.275	20	70
135	3.1875	0	70	105	3.15	10	70
145	3.2375	5	80	115	3.2	15	70
130	3.1625	0	70	115	3.2	20	70
145	3.2375	10	80	105	3.1	20	70
135	3.1375	10	80	65	3.133333333	15	70
95	3.183333333	5	70	120	3.15	15	70
115	2.9125	5	70	110	3.05	15	70
120	2.8625	0	70	70	3.066666667	10	70
120	2.9125	5	70	95	3.075	15	70
125	2.8625	0	70	125	3.3125	5	70
110	3.0625	10	70	110	3.1125	10	70
120	3.1125	15	70	100	3.5	25	60
120	2.9875	5	70	100	3.3	15	60
130	3.0375	10	70	110	3.35	20	60
130	3.0375	15	70	100	3.425	20	60
120	2.9375	15	70	110	3.225	15	60
80	2.916666667	10	70	120	3.275	20	60
135	2.9875	10	70	110	3.325	10	70
125	2.8875	10	70	120	3.375	15	70
85	2.85	5	70	120	3.25	5	70
110	2.9125	10	70	130	3.3	10	70
115	2.9125	10	70	130	3.3	15	70

115	3.1125	15	70	120	3.2	15	70
120	3.3375	25	60	80	3.266666667	10	70
120	3.1375	15	60	100	2.975	10	70
130	3.1875	20	60	105	2.975	10	70
120	3.2625	20	60	105	3.45	10	70
130	3.0625	15	60	100	3.5	5	70
140	3.1125	20	60	110	3.55	10	70
125	2.9125	15	70	110	3.425	0	70
130	3.1625	10	70	120	3.475	5	70
140	3.2125	15	70	120	3.475	10	70
140	3.0875	5	70	110	3.375	10	70
150	3.1375	10	70	70	3.5	5	70
150	3.1375	15	70	100	3.35	5	70
140	3.0375	15	70	120	3.6	15	100
100	3.05	10	70	130	3.525	10	100
120	2.8125	10	70	100	3.366666667	5	100
120	3.3375	5	70	90	3.233333333	5	80
130	3.3875	10	70	50	3.35	0	70
130	3.2625	0	70	105	3.525	5	70
140	3.3125	5	70	120	3.6	15	100
140	3.3125	10	70	110	3.5	15	80
90	3.283333333	5	70	70	3.666666667	10	70
120	3.1875	5	70	110	3.3	5	80
125	3.1875	5	70	70	3.4	0	70
140	3.4375	15	100	115	3.55	0	70
150	3.3625	10	100	110	3.3	0	70
120	3.15	5	100	80	3.466666667	5	100
110	3.016666667	5	80	115	3.2875	0	70
70	3.025	0	70	105	3.45	0	70
105	3.525	0	70	115	3.4625	0	70
120	3.6	10	100	105	3.625	0	70
70	3.666666667	5	70	115	3.3875	0	70
80	3.4	0	70	100	3.55	0	70
125	3.55	0	70	110	3.6	5	70
120	3.3	0	70	95	3.4	10	70
90	3.466666667	5	100	100	3.35	5	70
115	3.3625	0	70	125	3.3125	10	60
105	3.525	0	70	110	3.475	10	60
115	3.5375	0	70	105	3.325	20	60

105	3.7	0	70	110	3.275	15	60
115	3.4625	0	70	110	3.575	15	60
100	3.625	0	70	120	3.625	20	60
110	3.675	5	70	115	3.475	10	60
95	3.475	10	70	130	3.55	20	60
100	3.425	5	70	120	3.45	20	60
125	3.3875	10	60	80	3.6	15	60
110	3.55	10	60	100	3.225	15	60
105	3.4	20	60	105	3.175	10	60
110	3.35	15	60	105	3.225	15	60
120	3.575	10	60	110	3.175	10	60
130	3.625	15	60	120	3.325	10	80
115	3.55	10	60	80	3.433333333	5	70
130	3.625	20	60	125	3.575	5	70
80	3.7	15	60	120	3.325	5	70
100	3.3	15	60	90	3.5	10	100
105	3.25	10	60	110	3.575	10	60
105	3.3	15	60	120	3.625	15	60
110	3.25	10	60	135	3.3375	0	70
90	3.433333333	5	70	120	3.5	0	70
135	3.575	5	70	120	3.3	5	70
130	3.325	5	70	115	3.2	0	70
100	3.5	10	100	120	3.2	0	70
120	3.575	15	60	120	3.1375	5	70
130	3.625	20	60	105	3.3	5	70
135	3.4125	5	70	135	3.325	5	70
120	3.575	5	70	125	3.225	5	70
120	3.375	10	70	85	3.3	0	70
115	3.275	5	70	100	3.4	10	100
120	3.275	5	70	90	3.266666667	10	80
105	3.375	10	70	50	3.4	5	70
135	3.4	10	70	120	3.575	15	60
125	3.3	10	70	125	3.35	5	70
85	3.4	5	70	130	3.325	15	80
100	3.5	15	100	90	3.433333333	10	70
90	3.366666667	15	80	100	3	5	80
50	3.55	10	70	60	3	0	70
115	3.35	0	70	135	3.275	10	70
120	3.325	10	80	125	3.45	5	70

80	3.433333333	5	70	105	3.333333333	0	70
90	3.1	5	80	100	3	0	70
50	3.15	0	70	70	3.1	5	100
125	3.35	10	70	95	3.375	0	70
115	3.525	5	70	105	3.425	5	100
95	3.433333333	0	70	95	3.5	5	70
90	3.1	0	70	110	3.3625	5	70
60	3.25	5	100	140	3.2625	5	100
115	3.2875	5	70	130	3.1625	5	80
100	3.0875	10	70	90	3.216666667	0	70
105	3.0375	5	70	95	3.575	10	70
120	3.1875	10	70				