UNIVERSIDAD PANAMERICANA
FACULTAD DE INGENIERÍA

Con estudios incorporados a la
Secretaría de Educación Pública

"Algoritmos de Optimización del acceso a Internet

basado en Redes Definidas por Software"

TESIS

QUE PARA OBTENER EL GRADO DE

DOCTOR EN INGENIERÍA

P R E S E N T A

MTIA. Lorenzo Miguel Elguea Fernández

ASESOR:
Dr. Félix Orlando Martínez Ríos

CDMX 2021

MTIA. Lorenzo Miguel Elguea Fernández (Candidato al Doctorado)

Algoritmos de Optimización del acceso a Internet basado en Redes Definidas por Software

Tesis dirigida por Dr. Félix Orlando Martínez Ríos

En los últimos tres años, con los avances en redes definidas por software, la cantidad de procesos que pueden realizar los equipos de red, tanto los routers como los switches, se ha incrementado en varios órdenes de magnitud. Esto es posible debido a que la capacidad de cómputo, tanto para análisis como para la toma de decisiones, ya no depende del equipo en sí, sino que esta capacidad se ha trasladado a equipos externos, comúnmente controladores, equipos que al utilizar componentes estándares de cómputo, permiten con un costo menor, tener mayor flexibilidad y capacidad.

Las implementaciones de estas plataformas, son más comunes en las redes inalámbricas, en donde un controlador, generalmente externo, se encarga de la seguridad, optimizar los canales WIFI y las potencias de las diferentes antenas, balancear el número de usuarios entre equipos cercanos y otras funciones que difícilmente se puede incluir en el mismo "Access Point".

En la actualidad, los controladores suelen utilizarse con mayor frecuencia en la parte de la red LAN, ya que traen grandes beneficios sobre todo en segmentos como los destinados a los grandes centros de datos, donde miles de servidores se conectan a diferentes redes virtuales.

Esta tesis propone un algoritmo controlador de redes WAN, con funciones muy específicas para optimizar el tráfico externo, que es el que tiene asociado la mayor latencia en las comunicaciones.

Dedicatoria

Dedico este trabajo y agradezco a Dios por todo lo que nos da continuamente: la vida, la familia, el trabajo, todo lo que nos hace feliz y lo que nos cuesta trabajo también.

A mi familia, a mi esposa Rosy y a mi hijo Miguel, a nuestros papás y hermanos. A mi tía María Luisa✞.

A todos mis profesores, que de alguna manera influyeron en lo que soy y cómo pienso.

Agradecimientos

A mi director de Tesis, el Dr. Félix Martínez por sus enseñanzas, su paciencia, y sobre todo por su amistad.

A mis compañeros en la UP, a Patricia Santa Anna, José Luis Hernández, y a todos en la UP que de alguna manera me han ayudado a ser mejor cada día, en especial a los de Tecnologías de Información, Pily, Iara, Luis Enrique, Carlos, Marina, Mónica, Lucy, sólo por mencionar algunos, les agradezco a todos.

A Pedro Creuheras por todo su apoyo en estos años, a José Inés Peiro✞ y a mis colegas de TI, César V. Hernández de UP GDL, Ernesto Jonnatan Arroyo de UP AGS y Alfonso Díaz T. del Cedros.

A todos mis compañeros de estudio en el Doctorado, en especial a Marco Galindo y a José Cruz por todo lo que aprendí de ellos. A Ernesto Javier Kirschner Velázquez y a Cesar Hernández Ramírez de Bestel por su apoyo.

A los rectores y exrectores de la Universidad Panamericana, en especial al Dr. José Antonio Lozano, Dr. Santiago García, Dr. José Manuel Núñez, MBA. Luis Bonner, por impulsar la búsqueda de la verdad, el compartir el conocimiento, el cuidado de los detalles, y por hacer de la UP una Universidad de Investigación.

Índice general

Capítulos

1. Introducción 5

1.1. Contexto 5

1.2. Problema 6

1.3. Objetivos 7

1.3.1. Hipótesis de Trabajo 8

1.3.2. Síntomas, Alcances y Limitaciones 9

1.3.3. Metodología de Trabajo 10

2. Antecedentes y Estado de la Cuestión 11

2.1. Historia y Clasificación de las redes 11

2.1.1. Necesidades de Redes e Inicios de Internet 11

2.1.2. Internet: Red de Redes 13

2.2. Protocolos de Redes 16

2.3. Características de las Redes Definidas por Software 20

2.3.1. Esquema general y arquitectura de SDN 20

2.3.2. Plano de Control 22

2.3.3. Plano de Datos 23

2.4. Estado actual de la optimización de Redes 23

2.4.1. Soluciones de SDN 26

2.4.2. Soluciones Para Redes WAN 27

2.5. Resumen del Estado Actual de la optimización de Redes 27

3. Protocolo de Puerta de Enlace de Frontera: BGP 29

3.1. Sistemas Autónomos, camino más corto y próximo salto 32

3.1.1. Caminos o Paths 32

3.1.2. Representación con Grafos 34

3.1.3. Próximo Salto o Next-Hop 37

3.2. Tipos de BGP 37

3.3. Mensajes BGP 41

4. Propuesta de Optimización e Implementación 44

4.1. Optimización de Redes 45

4.2. Nuevo método desarrollado para comparar rutas 46

4.2.1. Selección de la métrica más adecuada para optimizar las rutas 46

4.2.2. Intervalos de confianza para la estimación de la Latencia 48

4.2.3. Análisis de la Ruta 54

4.3. Descripción del algoritmo de mejor ruta 55

4.3.1. Comparativo de Rutas 57

4.3.2. Inyección de Rutas y selección del próximo salto 59

4.3.3. Persistencia de las Rutas y con el mismo número de saltos 61

5. Experimentación y Resultados 63

5.1. Experimento Factorial en redes WAN 63

5.1.1. Factor Proveedor en redes WAN 66

5.1.2. Factor Protocolo en redes WAN 66

5.1.3. Factor Saturación a en redes WAN 66

5.1.4. Factor Tamaño del paquete en redes WAN 67

5.1.5. Factor Distancia en redes WAN 68

5.1.6. Análisis del Experimento Factorial 71

5.2. Experimentos de análisis de varianza (ANOVA) y comparación con algoritmo propuesto . . . 71

5.2.1. Experimentos Anova en redes cercanas . . . 72
5.2.2. Experimentos Anova en redes lejanas . . . 77
5.3. Desarrollo de la aplicación para la optimización de rutas sobre BGP . . . 80
5.3.1. BGPHeader.java . . . 81
5.3.2. BGPMain . . . 81
5.3.3. BGPNotificationPacket . . . 82
5.3.4. BGPOpenPacket . . . 82
5.3.5. BGPOpenParameter . . . 83
5.3.6. BGPPathAttribute . . . 83
5.3.7. BGPSendUpdatePacket . . . 84
5.3.8. BGPStatus . . . 84
5.3.9. BGPUpdatePacket . . . 85
5.3.10. InetNetwork . . . 85
5.4. Resultados de Optimizar Rutas . . . 87
5.4.1. Resumen de Resultados de los experimentos de optimización . . . 90
5.4.2. Comparativo de la solución propuesta con OpenDaylight . . . 94
5.4.3. Comparativo de la solución propuesta con λBGP . . . 94
5.4.4. Comparativo de la solución propuesta con xBGP . . . 95

6. Conclusiones 96
6.1. Comprobación de Hipótesis . . . 96
6.2. Resultados obtenidos . . . 98
6.3. Divulgación de la Investigación y Aplicación en la Industria . . . 98
6.3.1. Trabajos futuros . . . 100

Anexos

A. Código fuente de los programas desarrollados 121
A.1. Código Java implementación BGP . . . 121

A.2. Código Java que implementa diferentes Listados de Rutas 160

A.3. Código Java que implementa diferentes REST API para ODL 183

A.4. Código Java Ping para diferentes proveedores de Internet 186

B. Resultados detallados del análisis de Rutas y Características del hardware de red utilizado 189

B.1. Router Universidad Panamericana 189

B.1.1. Análisis de rutas IPv4 desde la Universidad Panamericana 2019 189

B.1.2. Análisis de rutas IPv6 desde la Universidad Panamericana 2019 190

B.1.3. Análisis de rutas IPv4 desde la Universidad Panamericana 2020 193

B.1.4. Análisis de rutas IPv6 desde la Universidad Panamericana 2020 195

B.2. Router Metrored características y análisis de sus rutas 196

B.2.1. Router Metrored IPv4 197

C. Tipos de Redes 201

C.1. Tipos y características 201

C.2. Analogía de WIFI con SDN 203

C.3. Protocolos de Enrutamiento 204

C.4. Herramientas de Redes 206

D. Glosario 208

Índice de Tablas

Cuadro

2.1. Clasificación de redes físicas según su escala. 16

2.2. Ejemplos de soluciones con SDN para distintos tipos de redes. 26

2.3. Soluciones con SDN para redes WAN. 27

3.1. Evolución de BGP desde el año 1984 hasta 2019. 29

3.2. Tipos de BGP su definición y referencias de casos de aplicación. 38

3.3. Tabla con las principales características de los routers sobre BGP. 39

3.4. Mensajes del protocolo BGP y documento de referencia que lo describe. 42

3.5. Frecuencia de los atributos BGP en Update. 42

5.1. Factores seleccionados. 64

5.2. Resultados de las 32 pruebas del experimento factorial. 65

5.3. Resumen estadístico de mediciones de la latencia sobre IPv4 en redes cercanas. 72

5.4. Resultados del análisis Anova de latencias en IPv4 en redes cercanas. 73

5.5. Límites de confianza para dos rutas diferentes en IPv4 en redes cercanas. 74

5.6. Resumen estadístico de mediciones de la latencia sobre IPv6 en redes cercanas. 74

5.7. Resultados del análisis Anova de latencias en IPv6 en redes cercanas. 74

5.8. Límites de confianza para dos rutas diferentes en IPv6. 75

5.9. Límites de confianza para dos rutas diferentes en IPv6 en redes cercanas. 75

5.10. Varianza Mínima de ambos protocolos IPv4 e IPv6 en redes cercanas. 76

5.11. Varianza Máxima de ambos protocolos IPv4 e IPv6 en redes cercanas. 76

5.12. Resumen estadístico de la latencia sobre IPv4 en redes lejanas. 77

5.13. Anova de latencias en IPv4 redes lejanas. 77

5.14. Resumen estadístico de la latencia sobre IPv6 en redes lejanas. 78

5.15. Anova de latencias en IPv6 redes lejanas. 79

5.16. Resumen de las rutas con posibilidad de optimizar por ISP. 90

5.17. Resumen de las rutas optimizadas por ISP. 91

B.1. Sistemas Autónomos en México conectados a Metrored sobre IPv4 y cantidad de rutas anunciadas. 196

C.1. Características de las redes físicas según su alcance geográfico 202

C.2. Clasificación de protocolos de ruteo. 204

Listado de Código

3.1. Listado con tres rutas BGP IPv6. 33

3.2. Listado con dos rutas BGP IPv4. 33

3.3. Listado con tres rutas BGP IPv4 optimizadas. 33

3.4. Listado con tres rutas BGP IPv4 con Prepend 34

4.1. Listado de ejecución con dos rutas BGP IPv4. 58

4.2. Listado de ejecución con tres rutas BGP IPv4. 58

4.3. Listado de ejecución con dos rutas BGP IPv6. 58

4.4. Listado de ejecución con tres rutas BGP IPv6. 59

5.1. Latencia.java 70

A.1. BGPHeader.java 121

A.2. BGPMain.java 122

A.3. BGPNotificationPacket.java 140

A.4. BGPOpenPacket.java 143

A.5. BGPOpenParameter.java 145

A.6. BGPPathAttribute.java 147

A.7. BGPSendUpdatePacket.java 148

A.8. BGPStatus.java 153

A.9. BGPUpdatePacket.java 153

A.10.InetNetwork.java 159

A.11.Listados.java 160

A.12.Reporte.java 166

A.13.RevisaConsola.java 169

A.14.Ruta.java 173

A.15.util.java 174

A.16.GetPutDefault.java 183

A.17.Latencia.java 186

B.1. Resultados rutas UP IPv4 2019 189

B.2. Resultados rutas UP IPv6 2019 191

B.3. Resultados rutas UP IPv6 2020 193

B.4. Resultados rutas UP IPv6 2020 195

B.5. Resultados rutas Metrored IPv4 197

Índice de figuras

Figura

2.1. Marcas líderes en optimización de tráfico WAN según IDC 2019 13

2.2. Capas del modelo OSI vs TCP/IP. 17

2.3. Estadísticas de uso de IPv6 en Google 2009-2010. 19

2.4. Estadísticas de uso de IPv6 en Google 2010-2020. 19

2.5. Modelo de tres capas de SDN. 22

3.1. Diferentes estados de un router BGP 31

3.2. Grafo BGP con 3 rutas en IPv6 35

3.3. Grafo BGP con Prepend en IPv4 35

3.4. Grafo BGP con 2 rutas en IPv4 36

3.5. Grafo BGP con ruta optimizada 36

3.6. Ejemplo de red con una conexión iBGP y tres eBGP 40

3.7. Ejemplo de Reflectores y Clientes de BGP 41

3.8. Captura de paquete con el Atributo 14 usado para el Next-Hop en IPv6. 43

4.1. Mínimo y máximo de los grados de libertad de la prueba t de Welch. 50

4.2. Ejemplo de rutas de IPv4 en BGP. 61

5.1. Utilización del ancho de banda de AS18734 9:00 am. 66

5.2. Utilización del ancho de banda de AS18734 1:00 pm. 67

5.3. Percentiles de dos tamaños de paquetes: 100 bytes y 1200 bytes. 67

5.4. Resultados de los experimentos para obtener la latencia en función de la distancia. 68

5.5. Interacción fuerte de los factores Distancia y Protocolo. 69

5.6. Interacción fuerte de los factores Distancia y Proveedor. 69

5.7. Interacción débil de los factores Protocolo y Proveedor. 70

5.8. Interacción débil de los factores Tráfico y Protocolo. 70

5.9. Latencias en IPv4 para determinar los límites de confianza en redes cercanas. 73

5.10. Latencias en IPv6 para determinar los límites de confianza en redes cercanas. 75

5.11. Comparativo de latencias de los protocolos IPv4 Vs. IPv6 hacia Netflix. 76

5.12. Límites de confianza latencias en IPv4 en redes lejanas. 78

5.13. Límites de confianza latencias en IPv6 en redes lejanas. 80

5.14. UML de BGPHeader 82

5.15. UML de BGPMain 82

5.16. UML de BGPNotificationPacket 83

5.17. UML de BGPOpenPacket 83

5.18. UML de BGPOpenParameter 84

5.19. UML de BGPPathAttribute 84

5.20. UML de BGPSendUpdatePacket 85

5.21. UML de BGPStatus 85

5.22. UML de BGPUpdatePacket 85

5.23. UML de InetNetwork 85

5.24. UML de las clases principales de la aplicación 86

5.25. Mejora en Latencias antes y después de optimizar rutas. 87

5.26. Distribución de las rutas de 623 destinos IPv4 según BGP 89

5.27. Distribución de las rutas de 623 destinos IPv4 por latencia. 89

5.28. Estadísticas de Google Meet 93

5.29. Estadísticas de Audio en Zoom 93

5.30. Estadísticas de Video en Zoom 93

5.31. Ejemplos de PageSpeed con calificaciones de 14 y 96 94

6.1. Posibles rutas a optimizar por número de Sistemas Autónomos. 100

C.1. Esquema de canales en las redes WIFI 203

C.2. Diagrama que muestra los alcances de los protocolos de enrutamiento. 205

Resumen

En esta Tesis Doctoral abordamos la utilización de Redes Definidas por Software (SDN por sus siglas en inglés), y aplicamos estas nuevas tecnologías para la optimización de rutas en Internet.

Problemática

El Internet está formado por sistemas autónomos que se conectan entre sí, esto crea diferentes caminos, las rutas son los diferentes caminos entre diferentes sistemas autónomos. La latencia es el tiempo que toma un paquete de información en llegar del origen al destino utilizando las rutas existentes. La distancia entre el origen y el destino es el principal factor directamente relacionado con la latencia, por lo que esta tesis lo que busca es reducir el tamaño de las rutas analizando la topología y la información obtenida por los mismos routers y así poder disminuir la latencia. Las rutas más largas tienen mayor variabilidad y son más propensas a fallas. Adicionalmente no existen métodos prácticos y rápidos que permitan compara latencias por diferentes rutas, método que se propone también en esta tesis.

Justificación

Las redes presentan latencia (Tiempo que tarda en transmitirse un paquete desde que se genera hasta que se recibe), y el mayor tiempo se utiliza en las redes de área amplia. Es por eso, que la reducción de estos tiempos en esta parte de la red, es donde se obtienen mayores beneficios. En las redes de área amplia (WAN), es donde se encuentran las conexiones a Internet, y es donde en los últimos años, se están migrando los servicios, como almacenamiento, generación de contenido y otros.

Objetivo General

Se propone un nuevo método para optimizar el tráfico de datos de Internet para reducir la latencia basado en una selección dinámica de rutas sobre el protocolo BGP usando redes definidas por software. Los

parámetros que mediremos y utilizaremos para optimizar las rutas son: la latencia y el número de saltos entre sistemas autónomos.

Metodología

En el **capítulo 4**, se relaciona la latencia con la distancia y se demuestra que existe una relación lineal entre estos dos parámetros. Con esta base, al reducir la distancia que se recorre, también se reduce la latencia, por lo que esta tesis se enfoca en determinar que rutas se pueden acortar y como corregirlas.

Resultados

En **Capítulo 5**, Experimentación y Resultados, se explica el algoritmo del ruteo usando BGP para Internet, se muestra cómo se pueden determinan las rutas a optimizar en la práctica y los resultados numéricos de la reducción de la latencia al modificar estas rutas (Proceso que se conoce como inyección de rutas). También en este capítulo se trata a detalle la latencia, métodos para medirla y compararla en diferentes rutas. En la sección de resultados se muestra un resumen de los resultados obtenidos en diferentes redes reales. En el primer capítulo, **Introducción**, se encuentra el objetivo de esta investigación, los alcances, limitaciones y las hipótesis. En el **Capítulo 2**, El estado del arte y Soluciones Actuales, se detalla: la necesidad de esta tesis, la situación actual, el marco teórico, SDN, los diferentes tipos de redes y protocolos, definiciones y todo lo que se ha realizado con respecto a la optimización de rutas en Internet. El **Capítulo 3**, Protocolo de Puerta de Enlace de frontera (BGP por sus siglas en inglés), se definen las características, operación y funcionamiento del protocolo BGP que es el estándar utilizado para todo Internet. Por la importancia de este protocolo en esta investigación, se ubica en un capítulo separado, aunque forma parte de los protocolos de enrutamiento que se ven en el capítulo 2. En el **Capítulo 6**, Conclusiones, se muestran las siguientes líneas de investigación y se relacionan las hipótesis con sus comprobaciones.

Los cuatro anexos, A, B, C y D, muestran los códigos de programación elaborados, los resultados encontrados en la ejecución de los códigos, los diferentes tipos de redes y un Glosario.

Abstract

In this Doctoral Thesis, we address the use of Software Defined Networks, and we apply these new technologies to optimize routes on the Internet.

The Problem

The Internet is a joint of autonomous systems that connect different networks, which creates different paths, and the routes are the different paths between autonomous systems. Latency is the time takes for a packet of information to get from the origin to the destination. The distance between origin and destination is the main factor directly related to latency, so this thesis seeks to reduce the path's size by analyzing the topology and routers' information, reducing latency. Longer routes have more significant variability and are more prone to failure.

Additionally, there are no practical and fast methods that compare latency by different routes, as proposed in this thesis.

Justification

Networks have latency (The time it takes to transmit a packet from its generation to its reception) and the longest time used in wide area networks. The most significant benefits occur in wide-area networks (WANs) Internet connections when reducing this times; in recent years, services, such as storage, content generation, and others, are being migrated.

General objective

Is proposed a new method to optimize Internet data traffic, to reduce latency based on a dynamic selection of routes over the BGP protocol using software-defined networks. We will measure and optimize the routes using latency and the number of hops between autonomous systems as parameters.

Methodology

Chapter 4, latency is related to distance, which shows the linear relationship between these two parameters. On this basis, by reducing the length traveled, latency is also reduced, so this thesis focuses on determining which routes to shorten and correct them.

Results

Chapter 5, Experimentation and Results, explains the routing algorithm using BGP for the Internet and shows how to optimize the routes in practice can be determined and the latency reduction's numerical results when modifying these routes (A process known as route injection). This chapter explains latency and discusses methods for measuring and comparing it on different paths. The results section shows a summary of the results obtained in various real networks. In the first chapter, **Introduction**, one can find the research's objective, outreach, limitations, and hypotheses.

Chapter 2, The state of the art and Current Solutions are detailed: the need for this thesis, the current situation, the theoretical framework, SDN, the different types of networks and protocols, definitions, and everything that is has done regarding the optimization of routes on the Internet.

Chapter 3, Border Gateway Protocol (BGP) defines the characteristics, operation, and performance of the BGP protocol, which is the standard used for the entire Internet. Due to the importance of this protocol in this research, it is located in a separate chapter, although it is part of the routing protocols seen in Chapter 2. **Chapter 6**, Conclusions, show the following research lines, and the hypotheses are related to their tests.

The four annexes, A, B, C, and D, show the programming codes developed, the results found in the code's execution, the different types of networks, and a Glossary.

Capítulo 1

Introducción

1.1. Contexto

Internet ha cambiado mucho. De las primeras redes con pocos equipos en la década de los 60s, hasta convertirse en servicios para miles de millones de usuarios en la actualidad, pasando por los equipos que proporcionaban servicios como Archie y Gopher, con base en archivos de texto y conectados por módems, este crecimiento requiere mejores herramientas y nuevos algoritmos para hacer más eficiente la comunicación entre equipos conectados a Internet.

Las redes de datos en los últimos años han evolucionado mejorando servicios, ancho de banda y han agregado nuevos protocolos. Un ejemplo de esto es el uso cada día más común del protocolo IPv6, que propone solucionar muchas de las deficiencias de IPv4 y permite, utilizando un esquema dual, tener redundancia de comunicación en muchos escenarios.

La seguridad, permite garantizar la comunicación entre dos equipos cifrando la información que atraviesa la red. Las VPN y demás tipos de túneles que permiten la conexión de clientes a redes privadas utilizando redes públicas como son el Internet. Las redes de área amplia, las redes que utilizan MPLS y Lan2Lan, así como las redes inalámbricas también son ejemplo de esta evolución de las comunicaciones de datos RFC4301 [Editor-RFC, 2019a].

Paralelamente, el crecimiento de usuarios, dispositivos y aplicaciones, plantean nuevos retos. Tendencias como el Internet de las cosas (que propone que dispositivos cotidianos se conectan a Internet, para dotarlos de la conectividad necesaria para mejorar su interacción con los seres humanos) imponen una necesidad de mejora de las comunicaciones, aprovechamiento del ancho de banda y el eficiente uso de las diferentes rutas para lograr una conectividad global, siempre contemplando la seguridad, privacidad y el costo del manejo de esta información.

Como ejemplos de nuevos dispositivos, todos ellos conectados a Internet, tenemos los vestibles (En

inglés Wearable) que son dispositivos que utilizamos para vestir o como accesorios, con funciones de monitoreo de los signos vitales, mostrar avisos, entretenimiento o simplemente conectarse a Internet; ya sea directamente o utilizando en la mayoría de los casos el teléfono celular inteligente. Los smartwatchs que son relojes electrónicos con capacidades de conectividad Bluetooth y WIFI, se utilizan mediante la sincronización con los teléfonos inteligentes como dispositivos para recibir notificaciones, acceder a algunas funciones del teléfono mediante instrucciones de voz y acceder a contenido en Internet. Por último, los HoloLens o lentes de realidad virtual de Microsoft, cuenta con cámara y mezclan imágenes virtuales con la visión de la realidad.

La necesidad de estas redes, que cada vez cubren zonas más amplias, llegando a más personas en todo el mundo, requieren mayor confiabilidad, disponibilidad y eficiencia, lo que implica una alta demanda de conectividad, inexistentes hace algunos años.

1.2. Problema

Los dispositivos de redes, principalmente los routers, son equipos que presentan algunas limitaciones respecto a los recursos que pueden destinarse a determinar mejores rutas. El procesador, la memoria y el sistema operativo, están dimensionados para soportar los protocolos estándares de redes. Esto puede no ser suficiente si se quiere asegurar que las rutas con las que trabaja el equipo, sean las óptimas respecto a la latencia, o algún otro parámetro como puede ser el costo o el balanceo de los enlaces.

Adicionalmente un factor que no consideran los protocolos actuales, es que las configuraciones de los equipos pueden presentar deficiencias que repercuten en la forma en la que se trasmiten y propagan las rutas en Internet. La fuente de rutas no optimizadas es el ser humano, en el sentido de que las omisiones en el anuncio de rutas son debido a fallas en las configuraciones de los routers, como es el caso de filtrados innecesarios hacia algunos vecinos de BGP, pues es difícil determinar cómo una configuración impacta en todas las rutas y equipos conectados. Esta es una consecuencia de la descentralización de la administración de Internet, ya que no se tienen mecanismos de validación en otros puntos de la red, sólo se tiene la información local [Amini et al., 2002], que muchas veces no es una representación real de las rutas en otros equipos. La optimización es esta Tesis, encuentra de manera automática este tipo de faltantes en las rutas recibidas y generalmente la propuesta es la inyección de esa ruta faltante y otros trabajos de investigación como [Hart et al., 2020, Wirtgen et al., 2020] también los abarcan.

Los problemas que se observan en Internet, se pueden clasificar en dos tipos:

• Información incompleta de la topología real de la red. Es común que los proveedores de Internet al tratar de disminuir la cantidad de rutas anunciadas, con la finalidad de requerir menos recursos en los routers, omitan el anuncio de algunos segmentos de red a algunos de sus vecinos. Estos vecinos, al no tener la información completa, la propagan con estas deficiencias a sus demás vecinos, lo que genera que existan algunos equipos que no conocen la "forma de la red" de manera precisa y sólo la aproximan con la información que reciben. En esta tesis solucionamos los problemas descritos anteriormente, al revisar exhaustivamente todas las rutas y construir la topología real, con la inferencia de la información de otros vecinos de nuestros equipos que, aunque no formen parte de la ruta hacia algún destino, pueden tener la información completa de la conectividad hacia ese destino o de algún punto intermedio para alcanzarlo.

• Ingeniería de Tráfico. El otro tipo de problema es la alteración de las rutas reales, con fines de ingeniería de tráfico que afectan el comportamiento de la red. Es común que los administradores de redes, al tratar de fomentar más a un proveedor que otro respecto a la entrada de tráfico en sus sistemas, utilice un método que altera las rutas de tal manera, que la latencia hacia su red se ve afectada. Generalmente esto ocurre con pequeñas redes que utilizan de manera exagerada la modificación de rutas. Igualmente, los errores al agrupar segmentos de red y la mala distribución de segmentos contiguos (como ocurre con IPv4), puede generar rutas que no sigan los caminos más cortos que realmente existen, incrementando con esto también el número de saltos entre los diferentes sistemas, haciendo esas rutas poco utilizables, aunque presenten una mejor latencia.

1.3. Objetivos

Esta Tesis Doctoral propone un nuevo método para optimizar el tráfico de datos de Internet, para reducir la latencia como **objetivo general** basado en una selección dinámica de rutas sobre el protocolo BGP usando redes definidas por software. Los parámetros que mediremos y utilizaremos para optimizar las rutas son: la latencia y el número de saltos entre sistemas autónomos. Este método consiste en remplazar algunas de las rutas de BGP, por otras de menor latencia, al reducir el número de Sistemas Autónomos (AS por sus siglas en inglés) identificando las rutas con un análisis completo de toda la información de los vecinos directamente conectados.

Generalmente esto se logra configurando rutas entre los diferentes equipos que proporcionan la conectividad entre los usuarios y los proveedores de contenido. Casi todas las rutas; ya sean estáticas, definidas manualmente, dinámicas, obtenidas mediante algún protocolo de ruteo como RIP o BGP; configuran los caminos que seguirá la información de manera general hacia algún segmento de red (Destino) sin considerar el tipo de datos que se desea trasmitir. La evolución de la tecnología de redes actualmente propone esquemas como SDN, donde la configuración de la red, es independiente del dispositivo que la implementa lo que brinda la posibilidad de elegir el mejor camino para la información que se transmite dependiendo del destino, del tipo de información y de otros parámetros.

La investigación se realizó dividiendo el problema en varios **objetivos específicos**:

1. Proponer una herramienta para comparar latencias en Internet que sea fácil de implementar.
2. Construir una aplicación para modificar las rutas en Internet compatible con los estándares actuales.
3. Definir un Algoritmo para determinar las rutas a modificar que mejore la latencia.

Para cada uno de estos objetivos, se evaluaron diferentes opciones. En las conclusiones se explican los motivos para la selección de las más apropiadas.

1.3.1. Hipótesis de Trabajo

Hipótesis 1. Considerando las diferentes rutas que se tienen en los routers, que se conectan a por lo menos dos proveedores de servicio de internet, ¿Se puede determinar si existe alguna ruta que mejore algún parámetro y que lo eficientice?

H1 El algoritmo "Analiza-Rutas" encuentra las mejores rutas en una red con base en la información de la topología completa y de los vecinos directos.

Se pueden ver los pasos de este algoritmo en la sección 4 (pág. 44) y el algoritmo en la sección 4.3.

Hipótesis 2. Considerando que existe un método para mejorar las rutas: ¿El método para mejorar las tablas de las rutas, es posible de implementar y funciona de forma eficiente?

H2 El método para mejorar las tablas de rutas en una red es práctico de implementar y mejora la latencia de las rutas optimizadas.

El resultado de la implementación de este método es la aplicación desarrollada que se encuentra en la sección 5.3 (pág. 80).

Hipótesis 3. Debido a que las métricas para optimizar rutas pueden depender de varios parámetros: ¿Qué parámetros son los que más reducen la latencia cuando se optimizan?

H3 La optimización de las rutas, entre el origen y el destino en una red depende del número de saltos entre sistemas autónomos; lo que reduce la distancia y a su vez, impacta en la disminución de la latencia.

Con el experimento factorial en la sección 5.1 (pág. 63) se relaciona la latencia con la distancia y se demuestra que los demás factores poco influyen, por lo que al reducir la distancia como se ve en la sección 5.2 (pág. 71) se mejora el tiempo de propagación. Adicional a la reducción de la latencia, hay una mejora en la varianza, como se ve en los experimentos en 5.4.1 (pág. 90).

1.3.2. Síntomas, Alcances y Limitaciones

Con esta investigación se optimizan rutas de Internet considerando conexiones o vecinos, con al menos dos proveedores y sin un número máximo de vecinos (Aunque se han evaluado en esta tesis redes con más de siete mil vecinos), ya que con un solo proveedor no se permite la selección de diferentes rutas. Se considerarán los dos protocolos de IP existente, que son IPv4 e IPv6.

La principal manifestación de rutas no optimizadas, suele ser que, en enlaces nacionales, cuya latencia debería de estar entre 4 ms. y 15 ms. (Ver sección 6.2), estos tiempos se elevan a más de 45 ms.

También se demostrará que, aunque existan más de 820,000 prefijos en IPv4 y 90,000 de IPv6 (Datos a octubre del 2020 [Chavez, 2010]), la necesidad de optimizar todos los prefijos no es importante. (Se tienen ejemplos como lo muestra la tabla 5.15 donde no hay beneficio de optimizar)

Debido a que el protocolo predeterminado de Internet es BGP, se utilizará este protocolo para obtener información de las rutas dinámicas. Adicionalmente, se mostrará el beneficio de esta optimización con algunas aplicaciones comunes dentro del ámbito académico y empresarial.

Algunos otros parámetros tales como: Rutas por proveedores más económicos o Políticas internas de

la compañía [Lorenz et al., 2017], Estrategias de uso de ancho de banda [Chase et al., 2014], Características especiales, como el uso de IPv4 o IPv6, Cancelación o castigo de rutas en IPv4 o IPv6 [Delling et al., 2009, Elguea et al., 2016] o Control de Accesos entre WIFI y LAN, no están contemplados en esta tesis pues solo nos enfocamos en el más importante que es la latencia como se demuestra con los experimentos factoriales.

Una de las principales limitaciones, [Liu and Sahabi, 2020], es que no se tiene control sobre el tráfico de entrada, solo el de salida, ya que solo este último permite seleccionar al proveedor de salida más adecuado y la entrada, depende de muchos más factores, sobre los que no se tiene prácticamente ningún control.

1.3.3. Metodología de Trabajo

Para realizar la propuesta y desarrollar la investigación creamos nuevo software y realizamos experimentos para:

- Definir un procedimiento para comparar rutas.
- Seleccionar qué parámetros se deben de optimizar para reducir el impacto en los equipos de redes que puedan generan errores tales como: perdida de paquetes, incremento de la latencia, etc.
- Aplicar las rutas que son más eficientes en los equipos de ruteo.
- Comparar de manera cualitativa nuestra solución con otras que se enfocan en otras optimizaciones como: compresión de la información, almacenamiento cache, selección del protocolo IPv4 o IPv6.

Adicionalmente y con la finalidad de definir experimentos y aprovechar los desarrollos, se generarán aplicaciones para la red de la Universidad Panamericana campus Mixcoac para lograr:

- Mejora de rutas con base a la latencia entre los tres proveedores actuales: AS6503, AS18734 y AS22566.
- Políticas internas: Firewall con información de la propia arquitectura de la red de la Universidad.
- Penalización de segmentos en IPv4 para fomentar el uso automático de IPv6.

Capítulo 2

Antecedentes y Estado de la Cuestión

Existen herramientas que permiten hacer más eficiente la operación de diferentes tipos de redes, por ejemplo, soluciones de SDN para redes locales (LAN, sus siglas en inglés) [Weng et al., 2018, Sun et al., 2019, Julong et al., 2019, Amiri et al., 2020], compresión de la información en redes amplias (WAN por sus siglas en inglés) [Nirmala, 2014, MacDavid, 2016] y controladoras para las redes inalámbricas (WIFI por sus siglas en inglés y/o LTE) que tiene la información de las interferencias de la red inalámbrica [Abbas et al., 2020, Guimaraes et al., 2020], para poder optimizar con base a estos datos. Hay estudios que muestran reconfiguración de redes dinámicamente [Auroux et al., 2015] y ya existen algunos algoritmos como rutas geométricas más cortas y optimización de redes [Mitchell, 2000], pero lo que propone esta investigación, es un método que proporciona más información a los algoritmos para tomar mejores decisiones.

Algoritmos más complejos, como Dijkstra aseguran la optimización de rutas, y se usan para redes internas, por ejemplo, las que utilizan el protocolo OSPF. Algunos autores como K. Krishna Chaitanya, proponen soluciones de BGP que buscan rutas usando OSPF [Chaitanya and Kishore, 2010] e incrementan los datos que manejan los routers.

2.1. Historia y Clasificación de las redes

2.1.1. Necesidades de Redes e Inicios de Internet

Las necesidades de las redes actuales, poco han variado de las necesidades de las primeras redes de datos: Eficiencia, Latencia, Disponibilidad, Confiabilidad son parámetros que siempre se han considerado en el diseño y desarrollo de las redes de comunicación.

Las redes en la actualidad se han vuelto más sensibles a problemas de latencia sobre todo en aplicaciones específicas tales como: el control remoto de actuadores, voz sobre IP, aplicaciones relacionadas con la información financiera en tiempo real, temas de Seguridad Nacional, clústeres de servidores, cómputo

paralelo, protocolos como el protocolo de sincronización de tiempo por red (NTP). Todas estas aplicaciones se beneficiarían con una reducción de la latencia.

También se ha incrementado la necesidad de más ancho de banda debido a aplicaciones como la transmisión de video, la transferencia de grandes archivos, la sincronización de bases de datos y prácticamente todo lo relacionado con contenido multimedia y en la actualidad ya es posible adecuar la red tomando en cuenta el tipo de aplicación y optimizarla para diferentes protocolos, destinos o tipos de usuarios que están conectados.

Tradicionalmente se ha separado la red en diferentes segmentos según las necesidades de optimización. También, los equipos de red han sido dimensionados o diseñados, prácticamente con una sola función principal: la de enviar los paquetes al destino. No estaba contemplada ninguna funcionalidad adicional, porque no se tenía la capacidad para eso o porque esa flexibilidad que daba el software, incrementaría mucho el costo de los equipos que componen la red.

Las redes WAN tienen más oportunidades de mejora en este aspecto, pues suelen ser las de menores recursos por lo que son el principal cuello de botella, las que más variables involucran y sobre las que se tiene menor control. También si consideramos que uno de los componentes que más incide en la latencia es la distancia de transmisión, cualquier algoritmo que reduzca la distancia que deben de viajar los paquetes, ayudará a reducir la latencia.

Adicionalmente, con soluciones como SDN, se puede extender las capacidades del router para hacerlo más “Inteligente” y esto permite tomar decisiones como la que propone esta tesis basada en el análisis detallado de las rutas.

Existen documentos con recomendaciones para cada segmento de red, por ejemplo, para las redes WAN, (donde sobresale el análisis que realizó Google en su propia red de área global, [Jain et al., 2013]), implementaron OpenFlow para maximizar el ancho de banda dividiendo el tráfico en múltiples caminos. Para lograr este objetivo, implementaron Switches personalizados y controlados centralmente para implementar la Ingeniería de Tráfico.

Existen soluciones comerciales para optimizar el tráfico entre diferentes centros de datos, como la de F5 [Kumar and Tech, 2014, F5-Networks, 2012], que utiliza equipos que encriptan la comunicación para hacerla más segura, y comprimen y deduplican la información (proceso por el que fragmentos de datos repetidos se

eliminan) para hacer más eficiente el tráfico y aprovechar mejor el ancho de banda basado en la calidad de servicio.

Soluciones de Firewall (Dispositivo de seguridad de red, que analiza el tráfico que pasa por el dispositivo y aplica políticas, permitiendo o negando el tráfico en función de reglas definidas) como las de Barracuda [Wu et al., 2013, Barracuda-Networks, 2013], implementan optimización de WAN, con soluciones como Cache, compresión y prioridades del tráfico.

La Figura 2.1 muestra el porcentaje de mercado de soluciones de este tipo, en donde sus productos tienen un enfoque destinado a oficinas remotas o centros de datos distribuidos, donde los ahorros en anchos de banda son reinvertidos para la compra de estos equipos. Estas soluciones, son parciales y no aplican para tráfico en Internet. Los principales fabricantes, Cisco, VMware, Silver Peak Nokia y Riverbed, tiene el 70 % del mercado. Estos productos son importantes, pues la forma en que optimizan, no se contrapone, al contrario, complementan la propuesta de esta tesis.

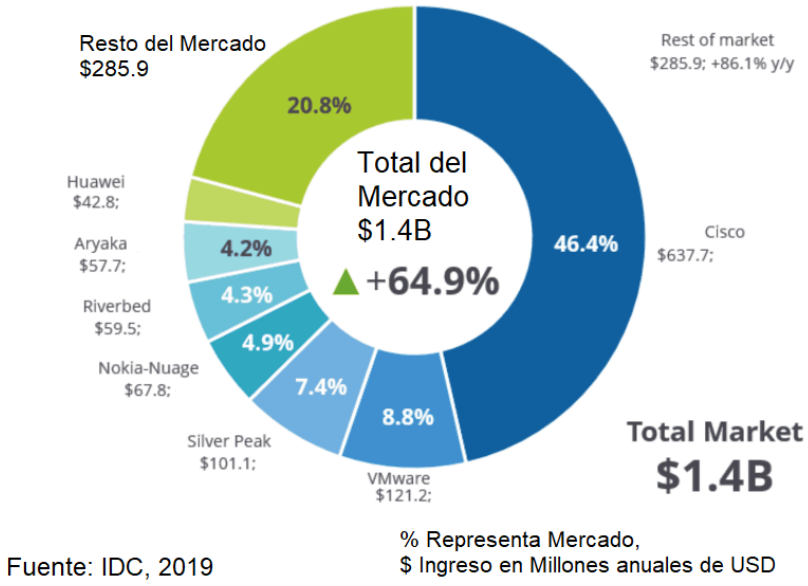

Figura 2.1: Marcas líderes en optimización de tráfico WAN según IDC 2019

[T4Labs, 2020]

2.1.2. Internet: Red de Redes

El día de hoy Internet es una infraestructura de información difundida, el prototipo inicial de lo que a menudo se llama la Infraestructura de Información Global. Su historia es compleja e involucra muchos

aspectos: tecnológicos, organizativos y relacionados con la comunidad de usuarios. Su influencia alcanza no sólo a los sectores técnicos de comunicaciones informáticas, sino en toda la sociedad a medida que avanzamos hacia el aumento de uso de herramientas en línea, para llevar a cabo el comercio electrónico, la adquisición de información y otras operaciones entre miembros de la comunidad. [Leiner et al., 2012b, p. 1] Lo mencionado en el párrafo anterior es relevante en la presente tesis por dos razones: la primera y más importante, el aspecto de la organización de Internet, donde el control no está implementado en ningún organismo y es la misma comunidad quien al unirse a la red agregan nodos adicionales, impidiendo que se tengan mecanismos impuestos de manera predeterminada para optimizar la red de manera local, por lo que la optimización actual se limita a ámbitos locales. Y la segunda razón, es referente a la evolución de la tecnología conforme al crecimiento de las redes, protocolos y servicios que se ofrecen.

Leonard Kleinrock en el MIT, publicó el primer artículo sobre la teoría de conmutación de paquetes [Leiner et al., 2012b, p. 2], en julio de 1961 y el primer libro sobre el tema en 1964. Kleinrock convenció a Robert J. Lefkowitz de la factibilidad teórica de comunicaciones a través de paquetes en lugar de circuitos, lo cual fue un gran paso a lo largo del camino hacia las redes de computadoras. El otro paso fundamental fue hacer que las computadoras utilizaran los mismos protocolos. Al explorar esto, en el año 1965 trabajando con Thomas Merrill, Robert conectó el ordenador TX-2 en la ciudad de Massachusetts al computador Q-32 en California con un dial-up de la línea telefónica de baja velocidad; creando así la primera red de ordenadores de área amplia. El resultado de este experimento fue la comprobación de que los equipos de cómputo de tiempo compartido podían trabajar bien en conjunto con los programas en ejecución y de recuperación de datos necesarios en la máquina remota, pero que la conmutación de circuitos del sistema telefónico era totalmente inadecuada para el trabajo. Se confirmó la hipótesis de Kleinrock de la necesidad de la conmutación de paquetes. A partir del concepto anterior surge el mecanismo de enrutamiento para enviar paquetes entre redes, buscando la mejor ruta. Este proceso de enrutamiento puede ser dinámico (se adapta a los cambios de topología) o estático.

Internet funciona mediante la conmutación de paquetes, estos son enviados por la ruta que proporciona el router que se encuentra conectado a los ISP, y éstos a su vez, envían estos paquetes a otros routers según rutas que se pueden determinar de manera estática (configuración manual) o de forma dinámica (protocolos de enrutamiento) siendo ésta última las más adecuada por tener la capacidad de adaptarse a los cambios de

topología para lo cual, utiliza el protocolo BGP de manera estándar en todo Internet. En redes conectadas con varios proveedores, como es el caso de la Universidad Panamericana, es recomendable implementar protocolos como BGP. Esto permite un control de las rutas, balanceo de tráfico y redundancia.

El aumento en el tamaño de Internet también desafió las capacidades de los routers. Originalmente existía un único algoritmo distribuido para el enrutamiento, implementado uniformemente por todos los routers de Internet. A medida que el número de redes en Internet crecía, este diseño inicial no podía expandirse atendiendo a las necesidades de crecimiento de la red; por lo que fue reemplazado por un modelo jerárquico de enrutamiento, con un protocolo de puerta de enlace interior (IGP por sus siglas en inglés) dentro de cada región de Internet y un protocolo de puerta de enlace exterior (EGP por sus siglas en inglés) utilizado para unir las regiones.

Este diseño permitió a diferentes regiones utilizar un IGP diferente, por lo que se podrían acomodar diferentes requisitos de costo, reconfiguración rápida, robustez y escalamiento. No sólo el algoritmo de enrutamiento, sino también el tamaño de las tablas de direccionamiento, hicieron necesario aumentar la capacidad de los routers. Recientemente se han introducido nuevos enfoques para la agregación de direcciones, en particular el enrutamiento interdominio sin clases (CIDR por sus siglas en inglés) [Leiner et al., 2012a], para controlar el tamaño de las tablas que son necesarias almacenar en los enrutadores.

Las redes, para su análisis, se suelen clasificar en dos categorías: Redes Físicas y Redes Lógicas. Las redes físicas, se clasifican según su tamaño y escala que abarcan, las redes lógicas, se pueden clasificar según los protocolos que utilizan. La Tabla 2.1, muestra en escalas sucesivas de potencias de diez los alcances de las diferentes redes. Es importante esta clasificación, pues las redes WAN son las que se ven afectadas por las diferentes rutas en Internet y son más sensibles a la distancia que es necesario recorrer para unir un nodo emisor con uno receptor, esto se puede ver en la Tabla C.1 (pág. 202).

Distancia entre procesadores	Alcance del Procesador	Ejemplo
1 m	$Metro^2$	Red de área Personal
10 m	Cuarto	Red de área Local
100 m	Edificio	Red de área Local
1 km	Campus	Red de área Local
10 km	Ciudad	Red de área Metropolitana
100 km	País	Red de área Amplia
1,000 km	Continente	Red de área Amplia
10,000 km	Planeta	Internet

Tabla 2.1: Clasificación de redes físicas según su escala.

Aunque menos conocidas, las redes tolerantes a disturbios (DTN por sus siglas en inglés) son un modelo de redes informáticas con un sistema de reglas para transmitir información [Suzuki et al., 2014, Jenkins et al., 2010, Wyatt et al., 2009, McMahon and Farrell, 2009], que extiende las capacidades de Internet terrestre a los ambientes de comunicación en el espacio exterior donde el Internet convencional no funciona bien. Estos entornos son típicamente sujetos a interrupciones frecuentes, enlaces que están limitados a una dirección, posiblemente largos retrasos y altas tasas de error.

2.2. Protocolos de Redes

Para reducir la complejidad de su diseño, la mayoría de las redes está organizada como una pila de capas o niveles [Tanenbaum, 2003, p. 26], cada una construida a partir de la que está debajo de ella. El número de capas, así como el nombre, contenido y función de cada una difieren de red a red. El propósito de cada capa es ofrecer ciertos servicios a las capas superiores, a las cuales no se les muestran los detalles reales de implementación de los servicios ofrecidos.

Estas capas se basan en el modelo de referencia de **Interconexión de Sistemas Abiertos** (OSI por sus siglas en inglés) que inició su desarrollo en 1977 y se publicó como estándar en 1984 y en el caso del modelo TCP/IP su relación de capas se muestran en la Figura 2.2. La consecuencia de tener las mismas bases que el modelo de capas OSI, permite la evolución y/o sustitución de capas, sin afectar el funcionamiento de toda la pila de protocolos. Con modelos así, la creación de protocolos, como IPv6, sólo conlleva realizar cambios en esa capa. El modelo OSI es un modelo teórico mientras que TCP/IP [Black, 1991, Briscoe, 2000, p. 8] es la implementación concreta del protocolo.

Internet Protocol v4 (IPv4 a partir de ahora) es el protocolo más ampliamente utilizado en Redes, se basa en el RFC760 y sus modificaciones. En 1969 la Agencia de Proyectos de Investigación Avanzada (Defense

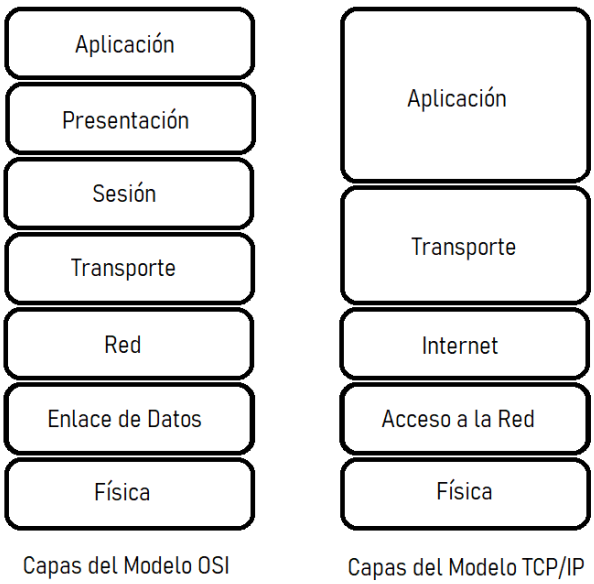

Figura 2.2: Capas del modelo OSI vs TCP/IP.

Advanced Research Projects Agency o DARPA) del Ejército de los EEUU desarrolla la ARPANET. En 1974 Se publica la primera norma TCP/IP, por lo que se puede decir que TCP/IP tiene más de 40 años. Su principal limitación es la cantidad de direcciones disponibles, por lo que se han creado mecanismos para permitir que se utilicen direcciones privadas, y éstas se conviertan a direcciones públicas mediante el proceso de traducción de direcciones de red (NAT por sus siglas en inglés) y que se encuentra definido en el RFC2663. Sus direcciones son de 32 bits.

Internet Protocol v6 (IPv6 a partir de ahora) es el protocolo de nueva generación, que promete resolver todas las deficiencias de IPv4, sobre todo en lo que respecta a cantidad de direcciones disponibles, y a esquemas de seguridad que fueron contemplados desde el origen del protocolo. Se basa en los estándares RFC1550, RFC1667, RFC1672, RFC1675, RFC1678, RFC1752 donde se le llamaba IP Next Generation y posteriormente cambió su nombre a IPv6 como en el RFC2460. Al capturar paquetes de IPv6, se puede ver que las direcciones, miden 128 bits.

Las principales diferencias entre IPv6 e IPv4 son:

- Direccionamiento de 32 bits en IPv4 vs 128 bits en IPv6.
- En IPv6 es más sencillo el direccionamiento ya que no utiliza clases como en IPv4.
- Encabezado con soporte para autenticación y encapsulamiento.
- Mejor seguridad con IPSEC.

- A diferencia de IPv4, en IPv6 no se requieren soluciones tipo NAT que afectan el flujo normal de Internet ya que la cantidad de direcciones es prácticamente ilimitada.
- En IPv6 no es necesario recalcular el campo de la redundancia, ya que no existe.
- Mejoras en Calidad de Servicio (QoS), además que se agregan etiquetas de flujo en el encabezado.
- Soporte para "Jumbo Frames" (paquetes de datos muy grandes) que mejoran la eficiencia, sobre todo en la transmisión de video (Video streaming) RFC2675.

La adopción de IPv6 fue lenta los primeros años, presenta un comportamiento exponencial, y en los últimos tres años pasó del cuatro al veinte por ciento. El segmento de la UP, 2801:f0:20:/48, se nos asignó el 8 de marzo del 2010. En los primeros años se utilizaban estrategias de adopción conocidas como túneles (Paquetes que encapsulan un protocolo sobre otro, por ejemplo, IPv6 en IPv4, los más comunes son Teredo, 6to4 e ISATAP que son implementaciones de diferentes tipos de túneles). La Figura 2.3 muestra la utilización de cada una de estas estrategias en los primeros años y la Figura 2.4 muestra el tráfico nativo de IPv6 en Google en los últimos siete años.

La adopción de IPv6 en el año 2010, no alcanzaba el uno por ciento y estaba compuesto por IPv6 nativo y por esquemas de transición como Teredo y 6To4. Aunque el crecimiento es exponencial, el tráfico en estas fechas aún no era tan notorio como se ve en la Figura 2.3 y en la Figura 2.4. Entre el año 2010 y el año 2017 continuó el crecimiento exponencial de IPv6. El tráfico en estas fechas sobrepasaba el veinte por ciento de los usuarios y prácticamente todo el direccionamiento es nativo, es decir con direcciones públicas de extremo a extremo de la conexión. El veinte por ciento de los usuarios tenía casi el treinta por ciento del tráfico en Internet ya sobre IPv6.

El protocolo BGP, por su importancia en Internet, y por ser la base de esta tesis, se tratará a detalle en el capítulo 3.

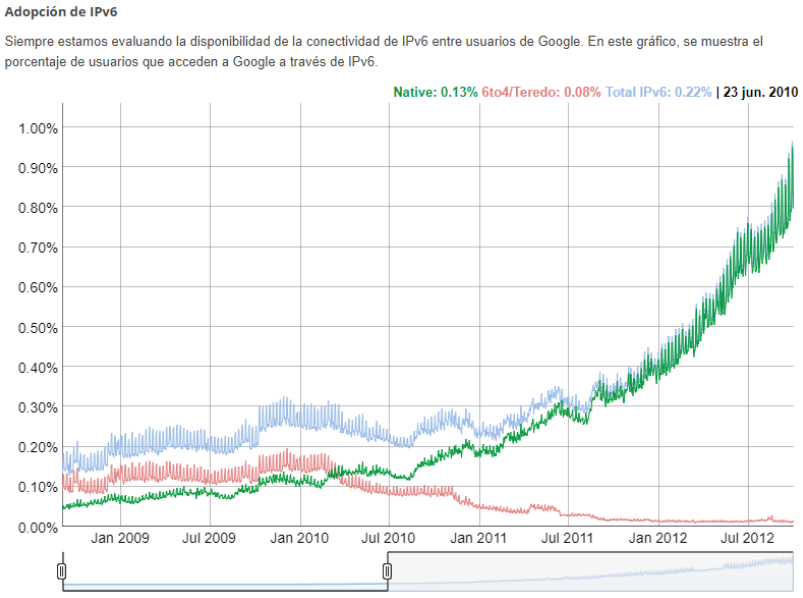

Figura 2.3:

Estadísticas de uso de IPv6 en Google 2009-2010. [Google, 2018]

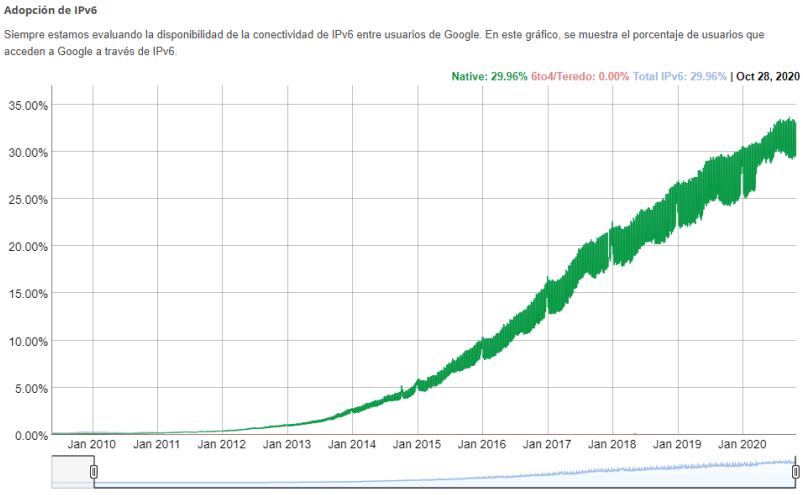

Figura 2.4:

Estadísticas uso de IPv6 en Google 2010-2020. [Google, 2020]

2.3. Características de las Redes Definidas por Software

En las SDN, un equipo externo a los equipos de redes, con capacidad de programación y de análisis, genera: las rutas, políticas y configuraciones de los equipos de red [Sezer et al., 2013, Shin et al., 2012, Voellmy and Wang, 2012] y es llamado el controlador de SDN. Esto facilita la separación en capas llamadas: plano de control y plano de datos. El controlador tiene la capacidad de modificar el plano de control. Esta propuesta genera nuevos retos, pero permite una mayor capacidad de control en los equipos. El lenguaje de programación que se utiliza en el controlador al ser un lenguaje moderno y de alto nivel (Java y Python) [Keshari et al., 2020], permite simplificar la codificación de las políticas o estrategias a implementar en los equipos de red además de que logra controladores altamente escalables.

2.3.1. Esquema general y arquitectura de SDN

Las SDN son una arquitectura que pretende ser dinámica, manejable, rentable y adaptable, tratando de ser adecuada para el gran ancho de banda y para la naturaleza dinámica de las aplicaciones actuales. Las arquitecturas SDN desacoplan el control de la red y las funciones de reenvío, para permitir el control de la red, lo que la convierte en un entorno programable para la infraestructura subyacente [Bruno Nunes et al., 2014]. Numerosos artículos mencionan los beneficios de esta arquitectura en donde se destaca principalmente la administración más fácil en ambientes de gran tamaño [Jarschel et al., 2013, Qazi et al., 2013, Cvijetic et al., 2013, Chen et al., 2015, Hernandez-Valencia et al., 2015].

Existen múltiples soluciones basadas en SDN, algunas sólo cubren la parte de la red LAN, como las soluciones propuestas por los fabricantes de switches: Cisco, Aerohive, Huawei, Extreme Networks, Juniper Networks, HP Enterprise e incluso Microsoft. Otras soluciones incluyen la parte inalámbrica y/o WAN, dentro de estas últimas, la opción de Cisco llamada Cisco OnePk con licenciamiento propietario. La más completa de las soluciones es Opendaylight con licenciamiento de código abierto.

El protocolo OpenFlow es un elemento fundamental para la construcción de soluciones SDN. Las características de la arquitectura de OpenFlow son:

1. Directamente programable: el control de red se puede programar directamente ya que está desacoplado de las funciones de reenvío, es decir, son módulos independientes y con características propias

cada uno.

2. Ágil: Esto es debido a que la abstracción de control de desvío permite a los administradores ajustar dinámicamente el flujo de tráfico de toda la red para satisfacer las necesidades cambiantes. Gestiona de forma centralizada: la inteligencia de la red es (lógicamente) centralizada en software basado en controladores SDN que mantienen una visión global de la red, que administra a las aplicaciones y a los motores de las políticas como un solo equipo de red lógico.
3. Configurable por Programación: pues permite a los administradores de red configurar, administrar, proteger y optimizar los recursos de red muy rápidamente a través de los programas de SDN dinámicos, automatizados, que pueden escribir los mismos administradores, porque los programas no dependen de software propietario.
4. Basado en estándares abiertos: Cuando se implementa a través de estándares abiertos simplifica el diseño de la red y la operación porque las instrucciones son proporcionadas por controladores SDN en lugar de múltiples dispositivos, proveedores específicos o protocolos.

Un plano o capa es una construcción conceptual que se asocia con una funcionalidad. En SDN, se divide en plano de control y plano de datos las dos funciones de los equipos.

En la Figura 2.5 se ve la separación entre las capas, lo que facilita delegar las tareas de cada parte a un proceso diferente, ya sea dentro del mismo equipo o en un esquema de equipo de red y controlador de SDN. El caso concreto de esta tesis, plantea un esquema router-controlador donde el controlador es una aplicación desarrollada en Java con capacidad de analizar y optimizar las rutas. Las tres capas de SDN: Infraestructura, Control y Aplicación, permiten la separación necesaria para dedicar procesamiento diferente para cada una de ellas, con funciones claramente definidas para cada capa.

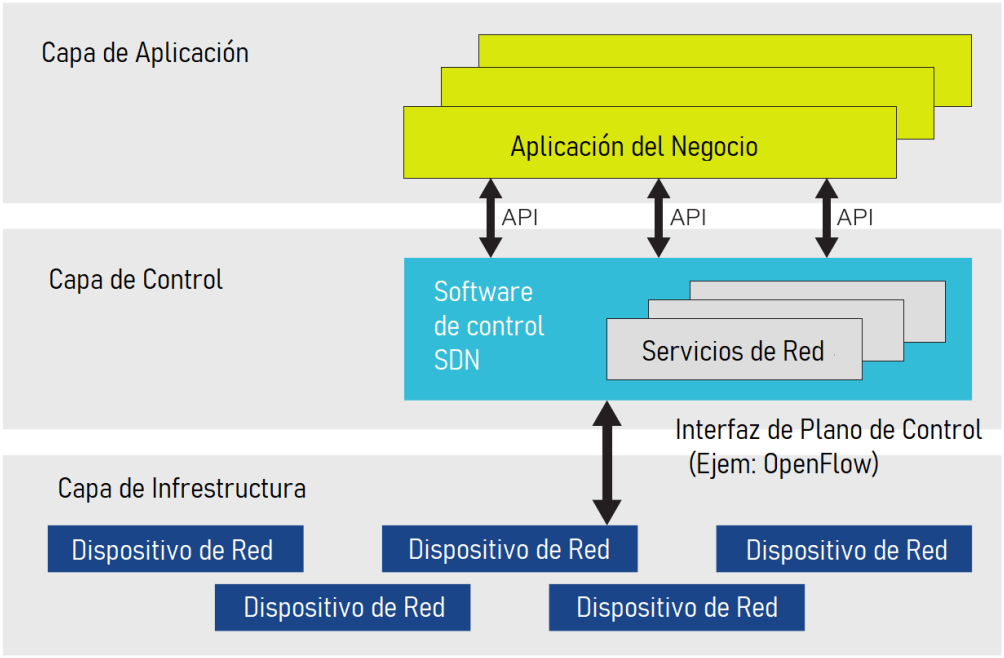

Figura 2.5: Modelo de tres capas de SDN.

[Brief, 2014]

2.3.2. Plano de Control

El router, como se mencionó anteriormente, es un equipo de red que determina la mejor ruta con base a protocolos de enrutamiento. El plano de control es la parte de la arquitectura del router que se ocupa de la elaboración del mapa de la red, o la información de una (posiblemente aumentada) tabla de enrutamiento que defina qué hacer con los paquetes entrantes. Funciones del plano de control (tales como la participación en los protocolos de enrutamiento) se ejecutan en el elemento de control.

En la mayoría de los casos, la tabla de enrutamiento contiene una lista de direcciones de destino y la interfaz de salida asociada con ellos. La lógica del plano de control también puede definir ciertos paquetes para ser desechados, así como diferentes tratamientos para las colas y calidades de servicios según las características del tráfico detectado. Principalmente el plano de control, maneja el tráfico que ocupan los equipos de red, para administrar, mantener y modificar el estado de la misma.

Dependiendo de la aplicación específica del router, puede haber una base de información de reenvío separada que se alimenta por el plano de control, y que se utiliza por el plano de reenvío para buscar paquetes, a muy alta velocidad, y decidir a donde enviarlos. [Shin et al., 2012, He et al., 2015]

2.3.3. Plano de Datos

El plano de datos (también llamado plano de reenvío) define la parte de la arquitectura del router que decide qué hacer con los paquetes que llegan en una interfaz de entrada. Más comúnmente, se refiere a una tabla en la que el router busca la dirección de destino del paquete entrante y recupera la información necesaria para determinar la ruta al elemento receptor, a través de la tabla de reenvío interno del router, y la interfaz de salida adecuada (posiblemente más de una).

En ciertos casos, la tabla puede especificar que un paquete va a ser desechado. En tales casos, el router puede devolver un mensaje "destino inalcanzable" (ICMP por sus siglas en inglés) u otro código adecuado. Algunas políticas de seguridad, sin embargo, exigen que el router debe descartar el paquete en silencio, con el fin de que un atacante potencial no se dé cuenta de que el objetivo está protegido.

Dependiendo de la implementación específica del router, la tabla en la que la dirección de destino se busca podría ser la tabla de enrutamiento (también conocida como la base de información de encaminamiento, RIB por sus siglas en inglés), o una base de información de reenvío separado (FIB por sus siglas en inglés) que es alimentada por el plano de control de enrutamiento, pero utilizado por el plano de reenvío de look-ups (Tabla de destinos muy eficiente usada en SDN) [Brief, 2014] a velocidades mucho más altas. Antes o después de examinar el destino, otras tablas se pueden consultar para tomar decisiones con base a otras características, como la dirección de origen, el campo identificador de protocolo IP o Protocolo de Control de Transmisión (TCP por sus siglas en inglés), User Datagram Protocol (UDP por sus siglas en inglés) o el número de puerto.

Las funciones de reenvío del plano de datos se pueden realizar en diferentes elementos físicos, estos componentes son llamados: elementos de reenvío. Los routers de alto rendimiento a menudo tienen múltiples elementos de reenvío distribuidos, de modo que el router aumente su rendimiento con el procesamiento paralelo. [Risdianto and Mulyana, 2012, Song, 2013]

2.4. Estado actual de la optimización de Redes

El rendimiento de las aplicaciones que requieren conexión a la red en tiempo real eficiente, la gestión de tráfico optimizado y la virtualización de recursos compartidos, ha acelerado la adopción de nuevos modelos de red. Las SDN, uno de los paradigmas de redes predominantes ha generado una cantidad importante de

investigaciones relacionadas con este tema. Tendencias como Internet de las cosas y temas relacionados con la seguridad de las redes, ha fomentado que las publicaciones y las patentes relacionadas con estos temas sean muy diversas y abundantes.

Existen patentes como las de Avaya en 2008 [Baldonado et al., 2008] en donde mecanismos externos, determinan la ruta más adecuada. Otra patente de Media Network Services en 2010 [Cicic and Bryhni, 2010], para un dispositivo que con base en las características del protocolo (en este caso Voz sobre IP) utilizando redes virtuales, seleccionan el dispositivo siguiente de envío de la información. La modificación en el año 2011 de la misma patente, en donde propone un esquema similar al de balanceo de cargas, pero con rutas y routers, para esto, se presenta un sistema y un método de operación de equipos y servicios para permitir un mejor transporte global de paquetes IP.

El gran uso de las redes WLAN, ha generado mucha investigación con respecto a la optimización. Algunos algoritmos [Kouhbor et al., 2006, Ergen and Varaiya, 2004, Eichenberger, 2006] se han enfocado al mejoramiento de la cobertura, el rendimiento y manejo de ancho de banda, en la movilidad y "roaming".

El caso de las redes WIFI, y redes de telefonía móvil [Abbas et al., 2020] utilizan un esquema de controladora-antenas, en donde la controladora toma decisiones centralizadas con base en la información de todas las antenas conectadas a ella. Esto beneficia la red en varios sentidos [Ergen and Varaiya, 2004, Cisco, 2015], el primero es que se optimiza globalmente, no localmente para cada antena (Por ejemplo, ajusta las frecuencias y potencias con base a todas las antenas y no sólo a una) y también por simplificar el control y la administración, lo que permite ahorros en operación y costos.

Inicialmente se pensó en proyectos de SDN para redes LAN, por lo que actualmente es donde existen más propuestas [Alba et al., 2004, Ephremides and Verdu, 1989, Kumar et al., 2002], como las que utilizan algoritmos híbridos y evolutivos para determinar mejores rutas. Existen artículos sobre el análisis del diseño de redes de comunicación en las que los métodos de control y la teoría de la optimización han demostrado ser útiles. Estos son el área de comunicación de acceso múltiple (para redes con enlaces compartidos, tales como redes de radio y redes de área local) y el área de enrutamiento de red (para redes con Interconexión punto a punto) y existen propuestas de diseño de topologías de red, usando un enfoque de algoritmos genéticos para minimizar latencias y costos de instalación.

Los centros de datos, con todos los servicios relacionados con redes LAN y WAN, proveen servicios

de almacenamiento, cómputo en la nube, virtualización y servicios de voz y de videoconferencias, no sólo en esquemas centrales, sino también distribuidos, que requieren optimizar todos estos servicios y manejarlos de manera eficiente. Una preocupación importante en los centros de datos es la eficiencia energética considerando los consumos eléctricos. Aunque este enfoque está fuera de los alcances de esta tesis, con soluciones como la propuesta aquí, en un futuro se podrán mover cargas de trabajo y procesamiento a lugares donde sean más eficientes [Beloglazov et al., 2012, Beloglazov and Buyya, 2010, Buyya et al., 2010, Greenberg et al., 2008], considerando variables como el costo de la energía.

En el cómputo en la nube existe una gran similitud con la optimización de las redes de datos, donde se busca el camino más corto entre dos nodos (En la LAN para el Centro de datos, y en la WAN en el cómputo en la nube). En realidad, este problema de optimización está solucionado desde hace varios años [S.B. Mitchell, 1998, Schreiber, 2007], no sólo para la red en un plano sino también en geometría esférica y otros esquemas 3D.

Las investigaciones más recientes, como las de Microsoft en 2015 [Garg and Bonaci, 2015], se han enfocado en proponer técnicas y sistemas para conectar múltiples routers en uno virtual, para implementar mecanismos de capa 1 y 2 en capa 3 (combinar nivel físico, de enlace de datos y la capa de red), y poder ejecutarlos en servidores.

A mediados del 2017, existían 6 aplicaciones, productos o proyectos que se pueden utilizar para la modificación de rutas BGP en los routers. El más difundido es OpenDaylight y sus características son:

- Es un proyecto Open Source.
- Patrocinado por las principales empresas de redes.
- Está desarrollado en Java.
- Cuenta con un módulo de BGP que permite mediante llamadas y actualizar rutas.

2.4.1. Soluciones de SDN

Las propuestas con base en SDN, abarcan diferentes ámbitos de las redes, desde las más comunes para redes LAN, y redes WIFI, y alguna solución para las redes WAN. Los más recientes (año 2018 y 2019) están relacionados con los siguientes temas:

Tipo	Descripción y características	Referencia
Redes WIFI	Redes de sensores inalámbricos definidos por software (SDWSN): Una revisión sobre recursos, aplicaciones y tecnologías eficientes	[Letswamotse et al., 2018]
Redes WIFI	Usar cualquier superficie para realizar un nuevo paradigma para las comunicaciones inalámbricas	[Liaskos et al., 2018]
Redes WIFI	Desactivar el cliente de Wi-Fi: decisiones más inteligentes utilizando una solución inalámbrica definida por software	[Saldana et al., 2018]
Redes WIFI	Presentación de ReCPRI: un protocolo reconfigurable en campo para la comunicación de retorno en una red de acceso por radio	[Vega et al., 2018]
Redes WIFI	Mitigación de amenazas de red mediante redes definidas por software para el Internet 5G de Radio Light System	[Cabaj et al., 2019]
Redes LAN	Software-Defined Data Center and Service Cluster Scheduling and Traffic Monitoring Method Therefor	[Wu and Shaofu, 2018]
Redes LAN	Equilibrio de carga mejorado en el enrutamiento de trayectos múltiples de igual costo basado en red definido por software en la red del centro de datos	[Dewanto et al., 2018]
Redes LAN	Análisis de rendimiento y evaluación de controladores distribuidos en red definidos por software en redes de centros de datos	[Muluye et al., 2019]
Otras Redes	Redes definidas por software en un sistema de televisión por cable.	[Kim et al., 2018]
Otras Redes	SDN-sniff: supervise el tráfico de múltiples clientes en la infraestructura de red virtualizada	[Fernández Sánchez, 2018]
Otras Redes	Laboratorios de seguridad para redes definidas por software en CloudLab	[Park et al., 2019]
IoT	Hacia un control de acceso tolerante a la indeterminación en Iot	[Heydari et al., 2019]
IoT	El impacto de las tecnologías de Internet de las cosas en la economía	[Tokareva et al., 2018]
IoT	Introducción a la sección especial sobre computación de borde para automóviles autónomos y conectados	[Pang et al., 2019]

Tabla 2.2: Ejemplos de soluciones con SDN para distintos tipos de redes.

2.4.2. Soluciones Para Redes WAN

Esta tesis se enfoca en optimización de las rutas de Internet, por lo que no todas las soluciones de SDN aplican. Las más enfocadas a redes WAN, son:

Descripción y características	Referencia
Optimización conjunta de la asignación de ancho de banda y la selección de rutas en ingeniería de tráfico compatible con QoE	[Wang et al., 2019]
Problemas de ubicación de controladores resistentes en redes de área amplia definidas por software	[Tanha, 2019]
Un mecanismo basado en BGP para el enrutamiento de menor costo	[Feigenbaum et al., 2005]
Reducción de la latencia de cola en almacenes de datos en la nube mediante la selección de réplica adaptable	[Suresh et al., 2015]

Tabla 2.3: Soluciones con SDN para redes WAN.

En general, las investigaciones sobre SDN, se han incrementado debido a la facilidad de implementar estas soluciones y la flexibilidad que se obtiene. Después de investigar sobre optimización de las rutas para mejorar la latencia, lo que propone esta tesis se considera innovador, pues no hay nada similar, los algoritmos actuales se enfocan más en memoria "cache", compresión, de-duplicación y/o balanceo de rutas para mejorar el rendimiento con base a una mejor utilización del ancho de banda.

2.5. Resumen del Estado Actual de la optimización de Redes

Con el desarrollo de herramientas como SDN y plataformas donde se separa el plano de datos del plano de control, se incrementan las posibilidades de manipulación de las rutas, siempre con una administración de la red más sencilla y automática. Esto se demuestra con los controladores de WIFI, el SDN aplicado a los centros de datos, donde facilita la virtualización de la red, el movimiento de las cargas de trabajo o el balanceo del tráfico. El otro aspecto importante a destacar es que no existen mecanismos eficientes para la optimización de rutas basados en la métrica de la latencia de la red.

En el año 2020 han aparecido publicaciones relacionadas con BGP y SDN. Dentro de estas, destacan λBGP [Hart et al., 2020] y xBGP [Wirtgen et al., 2020] por su similitud con la aplicación desarrollada. Adicionalmente, en septiembre del 2020 se liberó la versión "Aluminium"de OpenDaylight [Eftimie and Borcoci, 2020, Project, 2020b], (versión 13, ya que sus nombres representan elementos químicos desde el Hidrógeno, Helio,

hasta el Aluminio) Estas soluciones fueron comparadas con la propuesta de esta tesis, y aunque propiamente ninguna optimiza las rutas de BGP como el algoritmo aquí propuesto, son soluciones que pueden implementar algoritmos similares.

Capítulo 3

Protocolo de Puerta de Enlace de Frontera: BGP

Todo el Internet funciona mediante rutas aprendidas con el protocolo BGP. Con este protocolo se le informa a toda la red, porque ruta se puede llegar a un destino. Cada vez que se añade un segmento de red, este segmento es anunciado mediante BGP a todo el resto de la red, avisando en que sistema autónomo se encuentra y así se forma la ruta, primero con sus vecinos inmediatos, y luego los vecinos de los vecinos. También es importante este protocolo pues permite el filtrado y control de rutas. BGP es el protocolo predeterminado para anunciar rutas, es un protocolo de enrutamiento dinámico, de Gateway Exterior y de Vector Camino y al igual que los demás protocolos de enrutamiento dinámico, su función es la de mantener las tablas de rutas actualizadas para mostrar los mejores caminos en la red según la topología. También está clasificado como un protocolo de enrutamiento exterior de tipo vector ruta.

La historia de BGP inicia en 1984, con la especificación de EGP, antecesor de BGP. El primer documento [Lougheed and Rekhter, 1989] de BGP formal, se debe a Kirk Lougheed de Cisco Systems y Yakov Rekhter de IBM Corp. La Tabla 3.1 muestra cronológicamente su desarrollo, hasta mayo del 2019 existen 186 RFC relacionados con BGP [Editor-RFC, 2019b]:

Año	RFC	Descripción
1984	RFC904	Se crea formalmente la especificación del “Exterior Gateway Protocol”
1989	RFC1105 [Editor-RFC, 2019a]	Primera versión de BGP, por IBM y Cisco
1990	RFC1163 [Editor-RFC, 2019a]	Se genera la segunda versión de BGP
1991	RFC1267	Se libera la especificación para BGP-3
1994	RFC1654 [Editor-RFC, 2019a]	Primera versión de BGP-4
1995	RFC1771 [Editor-RFC, 2019a]	Segunda revisión de BGP-4
2006	RFC4271	Actualización de BGP-4
2007	RFC4721 [Editor-RFC, 2019a]	Se agrega Mecanismo de reinicio elegante para BGP
2007	RFC4760 [Editor-RFC, 2019a]	Se soporta la extensión multiprotocolo.
2009	RFC5492	Se agrega anuncio de capacidades
2014	RFC7313 [Editor-RFC, 2019a]	Capacidad mejorada de actualización de ruta en BGP-4
2017	RFC8203 [Editor-RFC, 2019a]	Comunicación de cierre administrativo de BGP
2017	RFC8205 [Editor-RFC, 2019a]	Especificación del protocolo BGPsec
2019	RFC8538 [Editor-RFC, 2019a]	Mensaje de notificación para reinicio elegante

Tabla 3.1: Evolución de BGP desde el año 1984 hasta 2019.

El protocolo BGP utilizado como estándar en Internet utiliza un algoritmo de distancia más corta

[Feigenbaum et al., 2005] con base en saltos entre sistemas autónomos. Esto es el equivalente a la analogía de calcular la distancia más corta en un mapa de carreteras tomando como parámetro el número de casetas. Esto no necesariamente es lo más adecuado, pues la distancia entre casetas no es constante, análogamente en Internet, la calidad de operación de los AS tiene mucha variación.

Cuando se utiliza un solo protocolo (generalmente IPv4), no existen muchas opciones con respecto a las rutas, pues se debe transitar por los diferentes AS intermedios, pero en el caso de tener "dual stack" (Ambos protocolos IPv4 e Ipv6 simultáneos), para IPv6 se pueden tener, adicional a las nuevas rutas, "atajos" utilizando túneles sobre IPv4, que hace más eficiente los enlaces entre los puntos requeridos.

La versión actual de BGP es "BGP 4", en la que se han incluido utilidades tales como: agregado de rutas, enrutamiento entre dominios sin clases y soporte multiprotocolo. Esta se encuentra definida en el RFC4271 y la primera versión en el RFC1771 [Editor-RFC, 2019a] y es la principal herramienta para la Ingeniería de Tráfico.

Su funcionamiento se basa en enviar el tráfico entre ASs. BGP intercambia información de enrutamiento para Internet y es el protocolo utilizado entre los ISP. Las redes de clientes, como universidades y corporaciones, generalmente emplean un protocolo de puerta de enlace interior (como RIP u OSPF), para intercambiar información de enrutamiento dentro de sus redes. Los clientes se conectan a ISPs, y los ISPs utilizan BGP para intercambiar rutas de clientes y de los ISPs. Cuando BGP se utiliza entre sistemas autónomos, el protocolo se conoce como BGP externo (eBGP). Si un proveedor de servicios está utilizando BGP para intercambiar rutas dentro de un sistema autónomo, el protocolo se denomina BGP interior (iBGP).

BGP es un protocolo de enrutamiento muy robusto y escalable, como lo demuestra el hecho de que es el protocolo de enrutamiento empleado por miles de proveedor en todo Internet. Para lograr la escalabilidad a este nivel, BGP utiliza muchos parámetros de ruta [Zhang and Bartell, 2003], denominados atributos, para definir directivas de enrutamiento y mantener un entorno de enrutamiento estable. Los vecinos BGP intercambian información de enrutamiento completa cuando se establece por primera vez la conexión TCP entre vecinos. Cuando se detectan cambios en la tabla de enrutamiento, los enrutadores BGP envían a sus vecinos únicamente las rutas que han cambiado. Los routers BGP no envían actualizaciones periódicas de enrutamiento y las actualizaciones de enrutamiento BGP anuncian sólo la ruta óptima a una red de destino.

El “Router ID” es el identificador del router, es un número de 32 bits que generalmente se representa como dirección IPv4. Este identificador, se utiliza para reconocer al router en las conexiones. Este identificador también se suele utilizar en OSPF y en EIGRP. En la mayoría de los sistemas de los routers, este valor si no se define, lo toma de las direcciones IPv4 de las interfaces del router.

La Base de Información de Enrutamiento (RIB por sus siglas en inglés) es una tabla con una selección de información de enrutamiento aprendida a través de protocolos de enrutamiento estático o dinámico. Los algoritmos utilizados en los RIB dependerán de los medios por los cuales BGP u OSPF determinan los mejores caminos.

El protocolo BGP utiliza seis diferentes estados en los que se encuentra el router con respecto a un vecino. La Figura 3.1 muestra los diferentes estados en los que se puede encontrar un router de BGP. El estado de «Established» es el que permite el intercambio de rutas en BGP. Con relación a estos estados, el de «Active» es común confundirlo con el de «Established» y un ruteador puede permanecer indefinidamente en un ciclo de «Connect» y «Active» y nunca realizar la conexión, si, por ejemplo, hay problemas que impiden alcanzar al router destino.

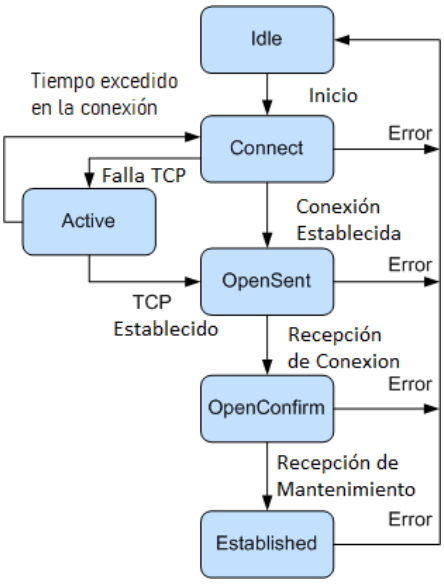

Figura 3.1: Diferentes estados en los que puede encontrarse un router [HUAWEI, 2013].

Estos estados fueron definidos inicialmente en 2 pliegos de una servilleta por Kirk Lougheed y Yakov Rekhter en una cafetería al finalizar una conferencia de “Internet Engineering Task Force (IETF)” [JABLONER, 2015] por lo que coloquialmente se le conoce como el protocolo de 2 servilletas.

Es importante aclarar que mantendremos los nombres originales de estos estados en inglés, pues representan los nombres propios de los estados y nunca son traducidos al español, incluso si usted trabaja sobre versiones configuradas para español en los dispositivos de red.

3.1. Sistemas Autónomos, camino más corto y próximo salto

Como ya hemos visto los AS son redes con políticas de rutas propias por lo cual un AS realiza su propia gestión del tráfico que circula entre él y los restantes ASs que forman Internet. Un número de AS (ASN por sus siglas en inglés) se asigna a cada AS que lo identifica de manera única. Estos AS son parte importante del protocolo de BGP, ya que la comunicación entre los diferentes sistemas autónomos se realiza mediante este protocolo y cada ASN identifica una red. Hay dos tipos de ASN disponibles: números AS de dos bytes y números AS de cuatro bytes. Un número AS de dos bytes está en el intervalo [1, 65535], mientras que un número AS de cuatro bytes pertenece al intervalo [1, 4294967295]. Los dispositivos que admiten ASN de cuatro bytes son compatibles con dispositivos que admiten ASN de dos bytes. Estos números son asignados por IANA (Internet Assigned Numbers Authority) [Stewart III, 1998, Zhao et al., 2001], a cada uno de los cinco RIR.

También se encuentran definidos unos segmentos para uso privado, es decir, no se debe de anunciar en Internet esto y son segmentos pertenecientes al intervalo [64512, 65534] en los ASN de dos bytes y en los ASN de cuatro bytes el intervalo es [4200000000, 4294967294]. Adicionalmente hay números reservados para propósitos especiales [Haas and Mitchell, 2014, IANA, 2014] y éstos se encuentran definidos en los RFC5398, RFC7607, RFC7534, RFC6996 y otros.

3.1.1. Caminos o Paths

Cada segmento de red es anunciado por un solo AS, generalmente éste es el AS al que pertenece dicho segmento. Ese AS anuncia ése y todos sus segmentos a los AS vecinos con los que tiene conexión. Los vecinos a su vez, anuncian su AS y los AS de sus demás vecinos a las conexiones restantes. Esto va formando el camino que debe seguir cualquier paquete en un AS origen hasta un AS destino.

El camino más corto es aquel por el que se recorren menos sistemas autónomos para llegar al destino. El tamaño del camino en IPv4 no tiene que ser igual al tamaño del camino sobre IPv6 para un mismo sistema

autónomo, ya que en IPv6 suelen ser más cortos. Un parámetro de referencia de la eficiencia de conectividad de un AS es el promedio del tamaño de las rutas que recibe.

Listado 3.1: Listado con tres rutas BGP IPv6.

```
1 00A-Aplus#sh bgp ipv6 uni 2A04:4E40::/48
2 BGP routing table entry for 2A04:4E40::/48, version 323279
3 Paths: (3 available, best #3, table default)
4    22566 3491 2914 54113
5    2001:1238::3:1 (FE80::3E8A:B002:2506:6F93) from 2001:1238::3:1 (201.161.6.115)
6    3549 3356 2914 54113
7    2001:450:2002:236::9D from 2001:450:2002:236::9D (67.16.168.99)
8    6503 174 2914 54113
9    2806:0:0:100::19 (FE80::6AEF:BDFF:FE58:F22) from 2806:0:0:100::19 (148.245.204.140)
```

El listado 3.1 muestra que existen 3 rutas (línea 3) para la red 2A04:4E40::/48 en el AS54113. Las rutas son: 22566 3491 2914 54113, 3549 3356 2914 54113, 6503 174 2914 54113, todas ellas parten del AS13679. En este caso, la mejor ruta es la 3, la que inicia en el AS6503, aunque las 3 rutas están formadas por 4 saltos o Sistemas autónomos intermedios.

Listado 3.2: Listado con dos rutas BGP IPv4.

```
1 00A-Aplus#sh bgp ipv4 uni 167.82.48.0/22
2 BGP routing table entry for 167.82.48.0/22, version 9515865
3 Paths: (2 available, best #2, table default)
4    22566 32098 6762 3356 3549 54113
5    201.161.34.97 from 201.161.34.97 (201.161.6.115)
6    6503 6762 3356 3549 54113
7    148.245.154.1 from 148.245.154.1 (148.245.204.140)
```

El listado 3.2 muestra que existen 2 rutas (línea 3) para la red 167.82.48.0/22 en el AS54113. Las rutas son: 22566 32098 6762 3356 3549 54113 y 6503 6762 3356 3549 54113. La mejor ruta es la 2, la que inicia en el AS6503, pues requiere 5 saltos en lugar de 6 que requiere la ruta que inicia con el AS22566

Listado 3.3: Listado con tres rutas BGP IPv4 optimizadas.

```
1 00A-Aplus#sh bgp ipv4 uni 167.82.48.0/22
2 BGP routing table entry for 167.82.48.0/22, version 11806601
3 Paths: (3 available, best #1, table default)
4 3549 54113
5 162.97.148.93 from 189.208.163.91 (1.2.3.4)
6 22566 32098 6762 3356 3549 54113
7 201.161.34.97 from 201.161.34.97 (201.161.6.115)
8 6503 6762 3356 3549 54113
9 148.245.154.1 from 148.245.154.1 (148.245.204.140)
```

El listado 3.3 a diferencia del 3.2 muestra, en la línea 3 que existen 3 rutas para la misma red 167.82.48.0/22. La ruta extra es la 3549 54113. Esta ruta es creada por la aplicación que optimiza ru-

tas, ya que detectó que el AS3549, [Hurricane Electric AS13679, 2020, Hurricane Electric AS22566, 2020, Hurricane Electric AS6503, 2020, Hurricane Electric AS3649, 2020], es un vecino directo del AS13679, por lo que se puede mandar el tráfico a este sistema, reduciendo el número de saltos a dos.

Algunos caminos BGP se agrandan intencionalmente repitiendo algún AS por los que se pasa. A esta manipulación del camino para agrandarlo (que se suele aplicar a tan sólo un vecino y no a todos) se le conoce como "Prepend" y tiene la finalidad de modificar la entrada ciertos vecinos para hacerlos menos óptimos. Podemos ver otros ejemplos en el Anexo B.1 o (18734 12989 23456x22 52821x7) que aparece en la red de Metrored para el segmento 138.121.146.0/24 Listado B.5 (pág. 197). Al número de ASs que existen en un camino también se le conoce como número de saltos entre AS y es la métrica más utilizada.

3.1.2. Representación con Grafos

Los caminos o Paths, se suele representar como grafos, para visualizar fácilmente las conexiones. El caso de BGP, todas las distancias se consideran unitarias (Solo se toma en cuenta el número de saltos) Las gráficas pueden crecer de manera muy rápida debido a la gran cantidad de conexiones, sobre todo de los principales ISP en EE UU. En las siguientes gráficas, Figura 3.2 y Figura 3.4 se muestra las rutas para los AS54113 en IPv6 y en IPv4 y representan las rutas mostradas en los listados 3.1 y listado 3.2.

El grafo de la Figura 3.5 muestra la ruta optimizada, que solo requiere de 2 saltos para alcanzar al SA54113, y la Figura 3.3 muestra el diagrama donde se aprecia el "prepend"que se hace para alargar el camino hacia el AS23596 del segmento de red 1.18.127.0/24 en el listado 3.4 (Líneas 4, 6 y 8) en el AS1237.

Listado 3.4: Listado con tres rutas BGP IPv4 con Prepend

```
1 00A-Aplus#sh bgp ipv4 uni 1.18.127.0/24
2 BGP routing table entry for 1.18.127.0/24, version 11368499
3 Paths: (3 available, best #3, table default)
4     22566 3491 174 4766 1237 1237 23596
5     201.161.34.97 from 201.161.34.97 (201.161.6.115)
6     6503 174 4766 1237 1237 23596
7     148.245.154.1 from 148.245.154.1 (148.245.204.140)
8     3549 3356 4766 1237 1237 23596
9     162.97.148.93 from 162.97.148.93 (67.16.168.99)
```

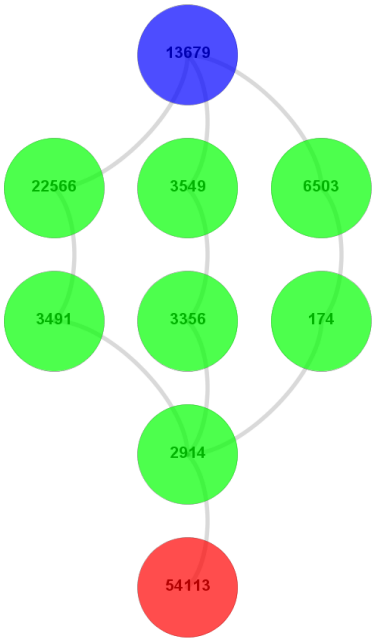

Figura 3.2: Grafo BGP con 3 rutas en IPv6

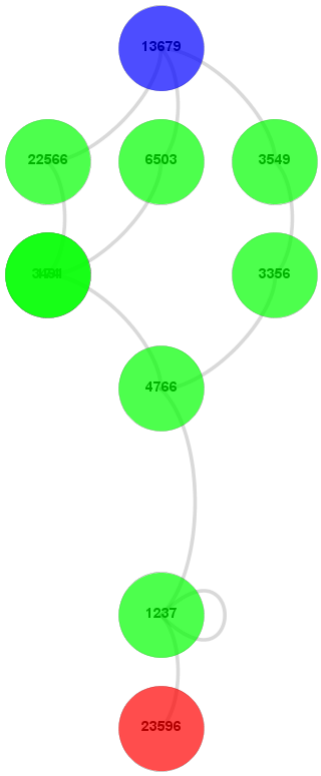

Figura 3.3: Grafo BGP con Prepend en IPv4

Los números representan sistemas autónomos.

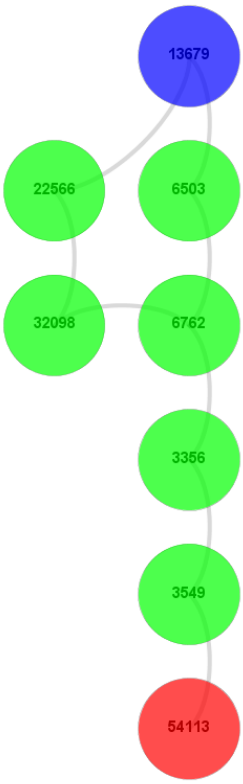

Figura 3.4: Grafo BGP con 2 rutas en IPv4

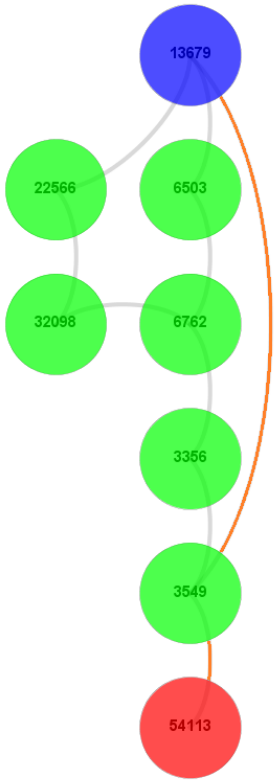

Figura 3.5: Grafo BGP con ruta optimizada

Los números representan sistemas autónomos.

3.1.3. Próximo Salto o Next-Hop

El próximo salto ("Next-Hop" por su nombre propio en inglés) es la dirección de red del dispositivo vecino al cual mandar los paquetes que se envían a ese destino. Es importante que el router pueda determinar la accesibilidad del vecino para el próximo salto, pues en caso contrario la ruta aprendida por BGP no estará correcta y no se adicionará a la tabla de rutas. Hay dos formas de resolver este problema: hacer accesible la dirección del próximo salto mediante otro protocolo (de rutas internas como RIP, IGRP, etc.) o modificando la dirección del próximo salto para sustituirla por una que el router conozca cómo acceder. En la práctica, las direcciones del Next-hop, suelen estar en el mismo segmento de red que las interfaces por las que se conecta el router al ISP, por ejemplo, en la UP las 3 direcciones del próximo salto en IPv4 son:

- 148.245.154.1 Mismo segmento que la interface g0/0 con ip 148.245.154.2 mask 255.255.255.248
- 201.161.34.97 Mismo segmento que g0/0/0 con ip address 201.161.34.98 mask 255.255.255.252
- 189.202.194.198 Mismo segmento que g0/2 con ip address 189.202.194.197 mask 255.255.255.252

y las 3 direcciones del próximo salto en IPv6 son:

- 2001:1238::3:1 Mismo segmento que la interface g0/0/0 con IPv6 2001:1238::3:2/126
- 2001:1278:1::F2 Mismo segmento que g0/2 con IPv6 2001:1278:1::F1/126
- 2806:0:0:100::19 Mismo segmento que g0/0 con IPv6 2806:0:0:100::1A/126

Este parámetro del próximo salto es muy importante en AS muy extensos pues las múltiples conexiones, incluso con el mismo vecino, requieren una dirección diferente para cada uno de los vecinos. La aplicación en Java, que desarrollamos en esta tesis, analiza y calcula los porcentajes de segmentos que tienen asignados los diferentes próximos saltos, y esta información es utilizada para agregar nuevas rutas que tiene como Next-hop una dirección diferente a la actual para así modificar la salida. Un ejemplo de modificación de rutas, consiste en agregar o "inyectar" una ruta nueva, en donde se cambia el valor de Next-Hop, para de esta forma alterar el proveedor que se va a utilizar.

3.2. Tipos de BGP

En la Tabla 3.2 podemos ver una comparación de los dos tipos de protocolos BGP: el interno y el externo, así como las referencias a sus especificaciones y las principales aplicaciones y consideraciones de utilización.

Es importante considerar que cuando se utiliza iBGP en más de dos routers, se debe tener una conexión de cada uno de los routers con todos los demás, para que todos los equipos cuenten con la misma información. En el caso en que esto no sea posible, se deben de utilizar equipos reflectores (Route Reflectors en inglés), que son dispositivos que se encargan de mantener informados a todos los equipos que se les conectan de las conexiones de la red.

Estos equipos reflectores se utilizan habitualmente para disminuir las conexiones iBGP, por ejemplo, si se tienen diez routers en el mismo AS, las conexiones que se requieren para conectarlos estarán dadas por la expresión $\frac{n(n-1)}{2}$ lo que implica que serían necesarias para nuestro ejemplo 45 enlaces, que permitirían la propagación de rutas entre los routers que componen nuestro AS.

Tipos de BGP	**Características y usos**
BGP Externo (eBGP): en este caso la conexión de los dispositivos es entre pares externos, cuando están en diferentes Sistemas Autónomos que el sistema local. Estas son las conexiones para el intercambio de rutas en todo Internet.	Las recomendaciones para usar eBGP están descritas en la siguiente referencia [Network Startup Resource Center, 2011]. Sus aplicaciones típicas son: intercambiar prefijos con otros ASs, implementar políticas de enrutamiento y permitir la creación de la "Ruta" utilizado el Sistema Autónomo.
BGP Interno (iBGP): en esta aplicación la conexión es entre equipos en el mismo AS. Las conexiones de iBGP se suele utilizar en AS muy grandes o que se encuentran físicamente distribuidos	Las recomendaciones para usar iBGP están descritas en la referencia [Network Startup Resource Center, 2011]. Las aplicaciones comunes para iBGP son: anunciar algunos o todos los prefijos de Internet a través de la dorsal, manejar los prefijos de los clientes y generalmente se recomiendan para redes de un mismo AS y que tienen un tamaño considerablemente grande como muestra la referencia de su aplicación.

Tabla 3.2: Tipos de BGP su definición y referencias de casos de aplicación.

Sobre el protocolo BGP los routers tienen diferentes nombres atendiendo a la función que realizan en el AS. La Tabla 3.3 podemos ver un resumen de éstos nombres y sus funciones dentro del AS. La Figura 3.6 muestra conexiones de BGP de ambos tipos, iBGP entre routers del mismo AS, el AS100 y eBGP entre el AS100 y el AS500, AS300 y AS400.

Un AS puede contener más de un router, entre los que se intercambian rutas mediante iBGP, y las conexiones hacia otros sistemas autónomos se realiza mediante eBGP. Las diferencias básicamente se dan en la forma en que se manejan los atributos de preferencia local. También otra diferencia está en el caso de varias rutas aprendidas algunas por iBGP y otras por eBGP [Teare, 2010], para este caso son prioritarias las rutas aprendidas por eBGP sobre las que se aprenden por iBGP; ya que en este caso se considera el parámetro de distancia administrativa.

Routers en BGP	**Caso de Uso**
Reflector de ruta (RR)	Un dispositivo BGP que puede reflejar (comunicar a los vecinos) las rutas aprendidas de un igual iBGP a otros pares iBGP.
Cliente	Un dispositivo iBGP cuyas rutas se comunican en el reflector de ruta a otros dispositivos iBGP. En un AS, los clientes sólo necesitan conectarse directamente al RR.
No Cliente	Es un dispositivo iBGP que no es RR ni cliente. En un AS, un cliente debe establecer conexiones de malla completa con el RR y con todos los demás no clientes. Una malla completa es cuando cada uno de los equipos se conecta con todos los demás.
Originador	Es un dispositivo que origina rutas en un AS. El atributo "Originator_ID" ayuda a eliminar los ciclos cerrados de enrutamiento en un clúster (conocido como bucles). Se suelen usar para centralizar el origen de las configuraciones en un solo equipo.
Cluster	Es un conjunto de RR y clientes. El atributo "Cluster_List" ayuda a eliminar los bucles de enrutamiento entre los clústeres. Este conjunto permite aplicar configuraciones similares a todos los equipos del AS.

Tabla 3.3: Tabla con las principales características de los routers sobre BGP.

La Figura 3.6 muestra conexiones de BGP de ambos tipos. Podemos ver conexiones iBGP entre routers del mismo SA, el AS100 y de sobre eBGP entre el AS100 y el AS500, AS300 y AS400.

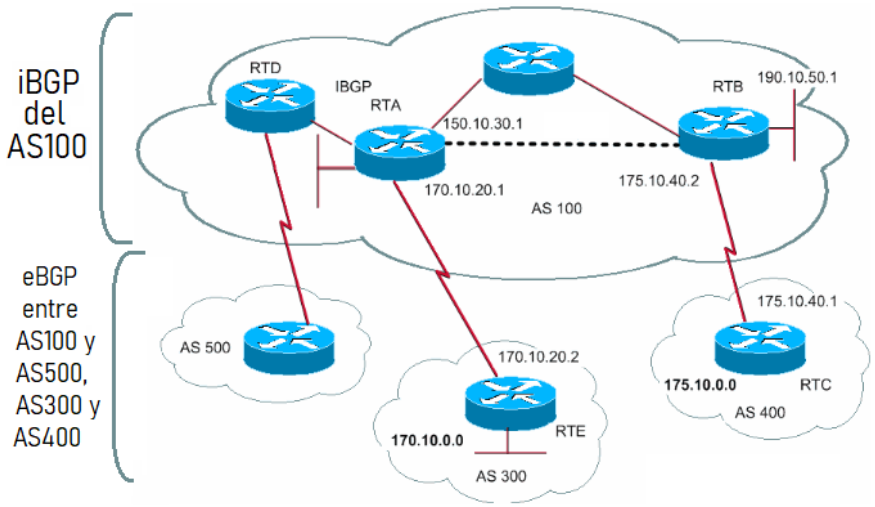

Figura 3.6: Ejemplo de red con una conexión iBGP y tres eBGP [Cisco, 2008].

En esta tesis, aunque las rutas que se modifican son las aprendidas por eBGP, la conexión se realiza mediante iBGP, ya que esto facilita la manipulación de esas rutas por pertenecer al mismo AS. Es recomendable que esta conexión se realice con un equipo directamente conectado en la misma red LAN, que el router, para de esta manera no depender de rutas externas, que, en el caso de fallar, pueden revertir todos los cambios realizados.

Al utilizar BGP como herramienta para inyectar diferentes rutas a los prefijos de red ya existentes, se logra que cualquier interrupción de la sesión de iBGP, restablezca las rutas a las condiciones iniciales, pues al desconectarse la conexión de BGP, esas rutas son automáticamente eliminadas de las tablas de ruteo.

En la implementación los Clientes se configuran normalmente con sus vecinos. En los Reflectores de Ruta se especifica que los otros routers son clientes (En Cisco se usa: route-reflector-client). En malla completa se requieren 32 conexiones para conectar los nueve routers clientes (para que todas las rutas se avisen a todos los equipos). Con Reflectores de BGP, se pueden reducir a 9 o 10 conexiones como se muestra en la Figura 3.7.

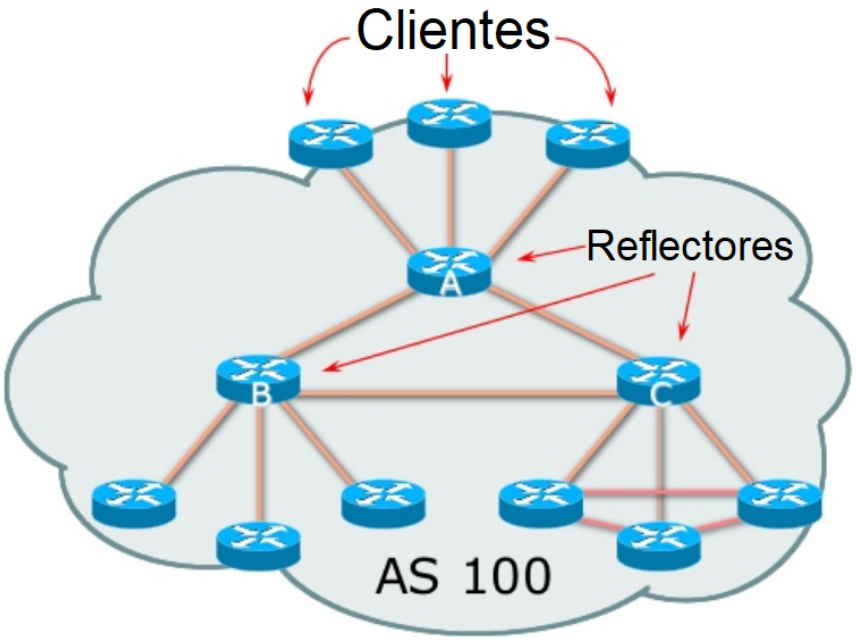

Figura 3.7: Ejemplo de RR y Clientes [Philip, 2016].

3.3. Mensajes BGP

Una vez que se ha realizado la conexión entre dos routers que utilizan BGP, el primer paso es la comunicación de las capacidades. Para lograr esto se intercambian mensajes con la información del protocolo. Estos mensajes se encuentran definidos en el RFC4271 y se describen en la Tabla 3.4. Los mensajes del protocolo son enviados según el estado de la comunicación. El mensaje de «Open» se usa sólo una vez, al iniciar la comunicación y los mensajes de «Update» y «KeepAlive» [Malik et al., 2019], se usan constantemente para indicar actualizaciones y mantener activa la comunicación respectivamente.

Una sesión típica de BGP en IPv4, envía los atributos mostrados en la Tabla 3.5. Durante las pruebas realizadas para esta Tesis Doctoral se encontraron incluso códigos obsoletos en algunas sesiones. Estos atributos son los que fueron detectados en una sesión IPv4 de BGP y además en la Tabla 3.5 son mostrados los porcentajes con los cuáles aparecen esos atributos. En la Figura 3.8 se muestra una captura de un paquete con algunos de estos atributos.

Valor numérico	Identificador del mensaje	Referencia que lo describe
0	Reserved	
1	OPEN	[RFC4271]
2	UPDATE	[RFC4271]
3	NOTIFICATION	[RFC4271]
4	KEEPALIVE	[RFC4271]
5	ROUTE-REFRESH	[RFC2918]
6-255	Unassigned	

Tabla 3.4: Mensajes del protocolo BGP y documento de referencia que lo describe.

Type Code value	Attribute Name	Conexiones	% de 106952
1	ORIGIN	106951	99.99 %
2	AS_PATH	106952	100 %
3	NEXT_HOP	106952	100 %
4	MULTI_EXIT_DISC (MED)	106952	100 %
5	LOCAL_PREF	106952	100 %
6	ATOMIC_AGGREGATE	8369	7.82 %
7	AGGREGATOR	14194	13.27 %
17	AS4_PATH	22660	21.19 %
18	AS4_AGGREGATOR	1300	1.22 %
21	AS_PATHLIMIT (deprecated)	4	0.004 %
32	LARGE_COMMUNITY	32	0.03 %

Tabla 3.5: Frecuencia de los atributos BGP en Update.

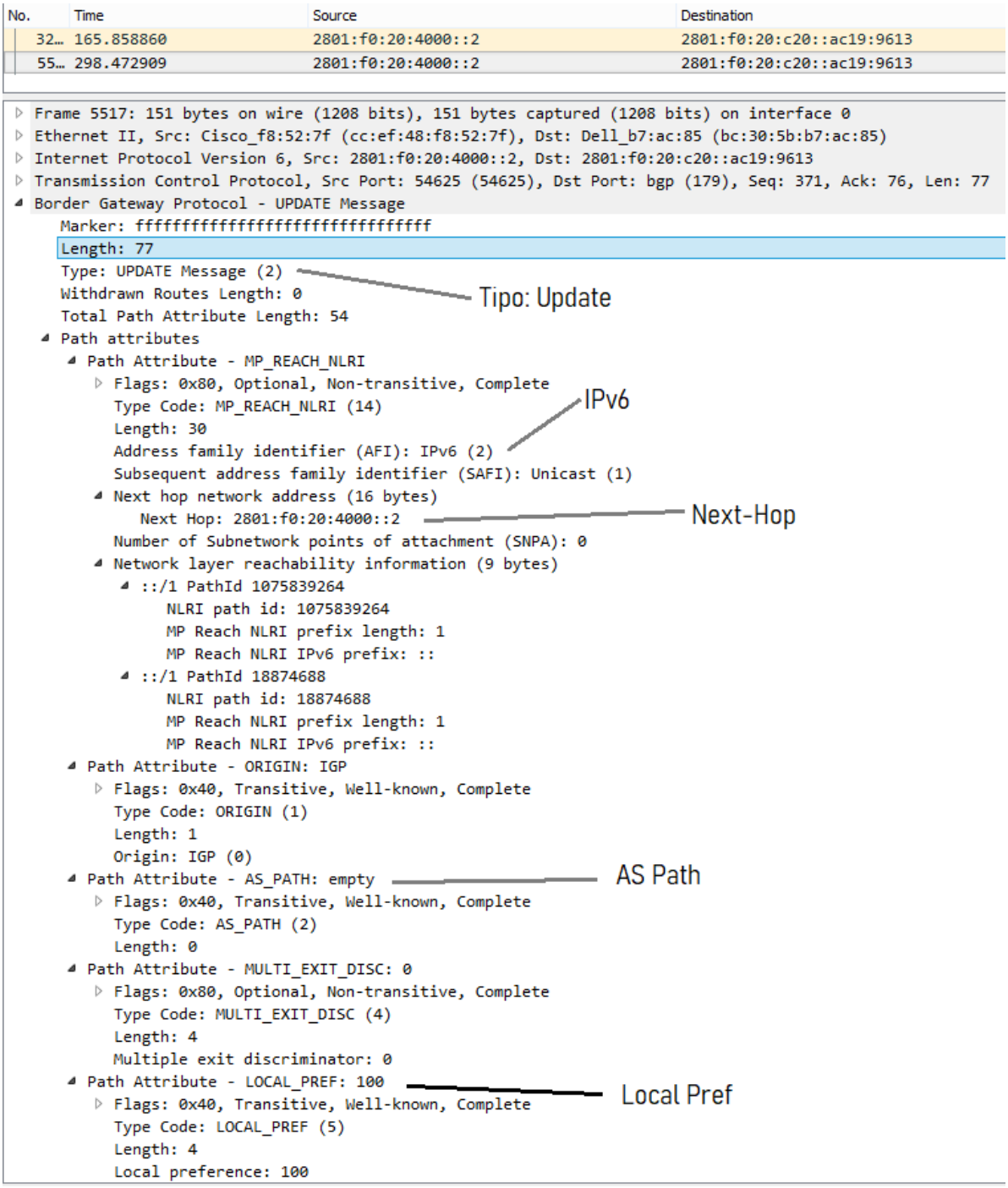

Figura 3.8: Captura de paquete con el Atributo 14 usado para el Next-Hop en IPv6.

Capítulo 4

Propuesta de Optimización e Implementación

En la investigación previa, se encontraron algunas herramientas para optimizar otros tipos de redes como son las LAN y WIFI (Ver sección C.1), solo recientemente se encontraron soluciones similares, como las que se mencionan en [Hart et al., 2020] y [Wirtgen et al., 2020], que no solo se enfocan a WAN, sino que también utilizan BGP. En el análisis de estas otras publicaciones se encontró, que no tenían métodos para comparar rutas, ni para optimizarlas. Para satisfacer estas dos necesidades, de gran importancia en redes actuales, [Eftimie and Borcoci, 2020] se desarrolló la propuesta de esta Tesis.

Esta propuesta, de manera general, busca reducir la latencia en Internet hacia algunos destinos de redes. Y como se demuestra en la sección 5.1 que la distancia es el factor de la latencia en redes WAN, se define que las métricas a evaluar son el tiempo de propagación.

También se propone un método para comparar latencias por diferentes rutas, con solo 5 mediciones y 3 parámetros (Latencia Mínima, Latencia Promedio y Latencia Máxima) utilizando Intervalos de confianza. Por último, se aplican los algoritmos siguiendo las etapas siguientes:

1. Analizar las rutas de Internet, para conocer toda la topología, en especial, lo que más interés representa, son los vecinos inmediatos o proveedores directos, pues son los que pueden repercutir en las latencias.
2. Buscar rutas que no estén optimizadas, es decir, aplicar el algoritmo propuesto para encontrar rutas en las que se pueda disminuir el número de saltos de sistemas autónomos y con ellos acortar las distancias, que son el principal factor de la latencia.
3. Arreglar las rutas, para utilizar los vecinos encontrados al optimizar las rutas.

4.1. Optimización de Redes

Desde la creación de las redes de cómputo siempre se ha tratado de que las comunicaciones sean óptimas, incluso en la primera red ALOHA se buscaba optimizar la utilización del medio en 1971, que, debido a colisiones, tenía un rendimiento apenas superior al 18 % utilizando el protocolo inicial llamado "ALOHA" y con una nueva versión de este protocolo denominado "Slotted ALOHA", se alcanzaba apenas el 36 %, y este protocolo, se sigue adecuando para redes de IoT [Vukobratović and Escribano, 2020] usando medios ópticos.

Recientes patentes como la de Cisco: "Intelligent host route distribution for low latency forwarding and ubiquitous virtual machine mobility in interconnected data centers"; la de Huawei Technologies "Compressed source routing encoding" y "Generating optimal pathways in software-defined networking" [Yang et al., 2015, Roch and Ashwood-Smith, 2015, Raps et al., 2015] indican que 45 años después continúa el proceso de optimizar las comunicaciones en las redes, incluso investigaciones recientes en el 2020, como la de [Liao et al., 2020] y tesis doctorales [Zhou, 2020] como la de Dr. Yi Zhou.

La aplicación propuesta en esta tesis, al cumplir con los estándares de iBGP, permite la conectividad con todos los ruteadores de Internet que manejan BGP, sin necesidad de renovación de hardware o la compra de licencias adicionales como ocurre con otras soluciones de SDN. Además, se puede inferir información de la red sin necesidad de que estrictamente esos parámetros sean obtenidos de los vecinos. Por ejemplo, podemos determinar algunos de los vecinos del destino o a qué países pertenecen los sistemas autónomos de los equipos que intervienen en toda la ruta. Esta flexibilidad no se ha encontrado en ninguna otra investigación hasta el año 2019.

Una ruta más corta significa una menor latencia. Especialmente en enlaces WAN, el principal factor que aumenta la latencia es la distancia física entre los dispositivos del enlace, como se comprueba en el experimento factorial 5.1 desarrollado en nuestra investigación.

El principal problema es que las condiciones de la red no se pueden anticipar, por lo que fue necesario desarrollar una herramienta para comparar las latencias durante esta investigación. Esta herramienta permite comparar las latencias de dos o más caminos y definir si hay beneficios significativos a la calidad de la comunicación si se modifican las rutas.

4.2. Nuevo método desarrollado para comparar rutas

La dificultad para comparar dos rutas en Internet se debe a dos factores: la topología de red cambia constantemente y en igualdad de latencia se prefieren rutas más estables, es decir con menor variación en la latencia. Esto implica que no se pueden evaluar durante mucho tiempo las rutas, pues se estarán midiendo variaciones debidas a los cambios de topología (además de los efectos del nivel de utilización de los enlaces en términos de su ancho de banda) por lo que se requiere un método rápido y que facilite comparar la latencia y su varianza.

4.2.1. Selección de la métrica más adecuada para optimizar las rutas

Para optimizar una red, lo más común es mejorar la ruta entre el origen y el destino. Para optimizar esta ruta, hay que determinar qué métrica se va a utilizar para definir si una ruta es mejor que otra, la métrica más utilizada es la de saltos entre ASs.

La poca homogeneidad de cada uno de los ASs en lo que respecta a alcance o capacidad instalada, dificulta que la métrica de menores saltos coincida efectivamente con la mejor ruta. Al utilizar dos o tres grandes proveedores, claramente puede ser mejor que usar uno o dos pequeños. Esto beneficia a las rutas de Internet al proveer caminos con menor latencia o de mayor ancho de banda.

Las métricas cualitativas (como las que utiliza BGP) suelen utilizar la ruta con el menor número de saltos entre routers, pero, sobre todo, rutas con base al camino con menor latencia. La latencia es el tiempo que demora un paquete en llegar del origen al destino. La latencia junto con el ancho de banda son los parámetros más importantes en el desempeño de una red.

Existe un protocolo especial sobre las redes IPv4 que ayuda a la medición de la latencia, el ICMP (sobre redes IPv6 se nombra ICMPv6) este protocolo de mensajes de control de Internet, permite obtener información relacionada con los mensajes de error de la red y del tiempo que se utiliza para llegar de un punto a otro.

El tiempo total es el período que el mensaje permanece en el medio (los enlaces y los equipos de red) hasta que lo recibe el destinatario. Se calcula como la suma del tiempo de propagación más el tiempo de transmisión más el tiempo de cola [Guo et al., 2015, Jamrozik et al., 1996, Gao et al., 1999, Banga et al., 1997], como se muestra en la Ecuación 4.1.

$$Tiempo\ Total = \text{Tiempo propagación} + \text{Tiempo trasmisión} + \text{Tiempo de cola} \tag{4.1}$$

Para determinar el Tiempo de propagación en la Ecuación 4.1 utilizamos la Ecuación 4.2 donde la "Longitud del enlace" es la distancia entre los equipos de red, y "Velocidad" es la rapidez de desplazamiento del mensaje que generalmente es una fracción de la velocidad de la luz atenuada por un índice de propagación o refracción según el medio. Para conexiones sobre Cobre se suele tomar un índice de 0.6 y para conexiones sobre fibra óptica se suele tomar 0.69

$$Tiempo\ propagación = \frac{\text{Longitud del enlace}}{Velocidad} \tag{4.2}$$

El "Tiempo de transmisión" en la Ecuación 4.1 se calcula con la Ecuación 4.3 donde el "Tamaño Paquete" se expresa en bytes o alguna de sus potencias y la "Tasa de Transferencia" en unidades de bytes por segundo.

$$Tiempo\ trasmisión = \frac{\text{Tamaño Paquete}}{\text{Tasa de Transferencia}} \tag{4.3}$$

El tiempo de cola depende del tipo y cantidad de equipos que retransmiten. Este factor está muy relacionado con la latencia que se produce en los Switches Ethernet. La latencia tiene dos componentes principales que se deben a variables deterministas y variables estocásticas. La latencia total en un Switch Ehternet, se debe a la acumulación de las latencias individuales de los siguientes componentes:

- Latencia en almacenamiento y reenvío de cada paquete
- Latencia en matriz de conmutación interna del Switch
- Latencia en el enlace con otros equipos
- Latencia en colas de paquetes

Las latencias en los Switches Ethernet [Anderson et al., 1993, Siemens Industry, 2014] y [Mace and Limited, 2014] suele estar en rangos de cinco a siete micro segundos como mínimo hasta 30-50 micro segundos como máximo. En una red LAN, donde puede haber hasta seis switches en línea, y distancias de entre 50 y 100 metros entre cada uno, las latencias en condiciones de tráfico normales son menores a 1 ms (En la práctica se pueden utilizar 0.015 ms. por cada switch).

4.2.2. Intervalos de confianza para la estimación de la Latencia

Los promedios de latencia para dos o más rutas, difícilmente nos pueden ayudar a determinar cuál de las rutas es la mejor. Esto es debido a que estrictamente este promedio no es suficiente ya que con las variaciones en las latencias no se tiene certidumbre respecto a si un promedio es mejor que otro. Para solucionar este problema lo correcto es utilizar intervalos de confianza para la diferencia de medias de dos distribuciones normales, con varianzas desconocidas, pero diferentes (esto se debe a que cada proveedor altera de diferente manera la varianza de cada camino o ruta al utilizar diferentes equipos de red y otros enlaces). La Ecuación 4.4 muestra como lo calcularemos [Yuen, 1974]:

$$t = \frac{(\overline{x}_1 - \overline{x}_2) - (\overline{\mu}_1 - \overline{\mu}_2)}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} \tag{4.4}$$

Esta estadística de la prueba t de Welch, tiene ν grados de libertad donde:

$$\nu = \frac{(\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2})^2}{[\frac{(\frac{\sigma_1^2}{n_1})^2}{(n_1-1)}] + [\frac{(\frac{\sigma_2^2}{n_2})^2}{(n_2-1)}]} \tag{4.5}$$

Como ν no es un número entero, se truncará su valor por redondeo al entero menor más cercano [Andreev and Kanto, 2004] (ya que los grados de libertad reflejan el número de parámetros libres, se puede aproximar al entero menor más próximo). No se debe confundir este valor con el del estimador combinado de la desviación estándar común de la población con $n_1 + n_2 - 2$ grados de libertad.

Despejando de la Ecuación 4.4 la diferencia de medias poblacionales nos queda:

$$\overline{\mu}_1 - \overline{\mu}_2 = (\overline{x}_1 - \overline{x}_2) \pm t\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}} \tag{4.6}$$

Si se asume que para cada ruta se realizan el mismo número de pruebas se tiene que $n_1 = n_2 = n$ y se puede simplificar la Ecuación 4.5 de la siguiente forma:

$$\nu = \frac{(n-1)\left(\sigma_1^2 + \sigma_2^2\right)^2}{\sigma_1^4 + \sigma_2^4} \tag{4.7}$$

De la Ecuación 4.7 podemos obtener el mínimo y el máximo para μ a través de las Ecuaciones 4.9 y 4.10.

$$f(\sigma_1, \sigma_2) = \left\{ \frac{\left(\sigma_1^2 + \sigma_2^2\right)^2}{\sigma_1^4 + \sigma_2^4} \right\} \tag{4.8}$$

$$min\left\{ \frac{\left(\sigma_1^2 + \sigma_2^2\right)^2}{\sigma_1^4 + \sigma_2^4} \right\} = 1 \tag{4.9}$$

$$max\left\{ \frac{\left(\sigma_1^2 + \sigma_2^2\right)^2}{\sigma_1^4 + \sigma_2^4} \right\} = 2 \tag{4.10}$$

$$\frac{\partial f}{\partial \sigma_1} = \frac{4\sigma_1\left(\sigma_1^2 + \sigma_2^2\right)}{\sigma_1^4 + \sigma_2^4} - \frac{4\sigma_1^3\left(\sigma_1^2 + \sigma_2^2\right)^2}{\left(\sigma_1^4 + \sigma_2^4\right)^2} \tag{4.11}$$

$$\frac{\partial f}{\partial \sigma_2} = \frac{4\sigma_2\left(\sigma_1^2 + \sigma_2^2\right)}{\sigma_1^4 + \sigma_2^4} - \frac{4\sigma_2^3\left(\sigma_1^2 + \sigma_2^2\right)^2}{\left(\sigma_1^4 + \sigma_2^4\right)^2} \tag{4.12}$$

De la ecuación 4.7 y los resultados de las ecuaciones 4.9 y 4.10 se puede determinar que los grados de libertad están acotados en el intervalo $[n-1,\ 2n-2]$, lo que simplifica los cálculos al no necesitar la tabla estadística de grados de libertad completa. En el caso de realizar 5 muestreos de la latencia, los grados de libertad siempre estarán entre 4 y 8 como se muestra en la Figura 4.1.

Supongamos que la función que se desea maximizar tiene cinco variables independientes (los 5 posibles tiempos de la latencia) cumpliendo con los mínimos muestreos del estadístico de Welch [Lee, 1992, Jan and Shieh, 2011], de ellos al menos dos valores son conocidos, el tiempo mínimo y el tiempo máximo, estos dos valores conocidos acotan los otros tres que no se conocen. Entonces de la Ecuación 4.13 de la varianza:

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2 \tag{4.13}$$

La ecuación anterior puede ser escrita nuevamente si sustituimos en la Ecuación 4.13 los siguientes valores: $n = 5$, $x_1 = x_{min}$ y $x_5 = x_{max}$ tenemos que la ecuación anterior sólo depende de x_2, x_3 y x_4 y podemos reescribir la varianza como:

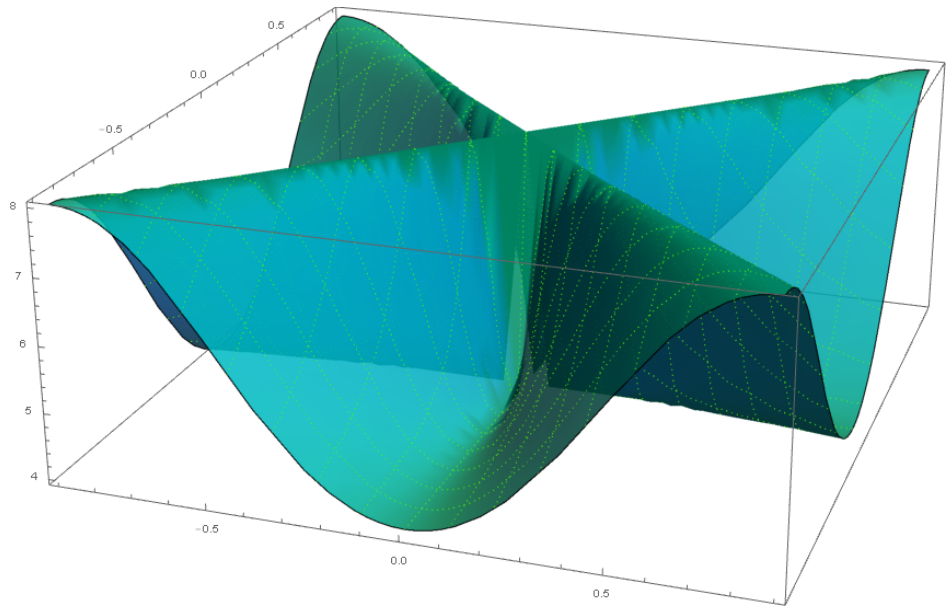

Figura 4.1: Mínimo y máximo de los grados de libertad de la prueba t de Welch.

Con 5 muestreos (n=5).

$$\sigma^2 = \frac{(x_{min} - \overline{x})^2 + (x_2 - \overline{x})^2 + (x_3 - \overline{x})^2 + (x_4 - \overline{x})^2 + (x_{max} - \overline{x})^2}{5} \tag{4.14}$$

La Ecuación 4.14 es la función que queremos maximizar sujeta a la restricción siguiente:

$$\overline{x} = \frac{x_{min} + x_2 + x_3 + x_4 + x_{max}}{5} \tag{4.15}$$

Utilizando Multiplicadores de Lagrange, y suponiendo que $f(x_2, x_3, x_4)$ es la función a maximizar sujeta a la condición $g(x_2, x_3, x_4) = c$, podemos escribir:

$$f(x_2, x_3, x_4) = \frac{(x_{min} - \overline{x})^2 + (x_2 - \overline{x})^2 + (x_3 - \overline{x})^2 + (x_4 - \overline{x})^2 + (x_{max} - \overline{x})^2}{5} \tag{4.16}$$

$$g(x_2, x_3, x_4) = x_{min} + x_2 + x_3 + x_4 + x_{max} - 5\overline{x} \tag{4.17}$$

Resolvemos éste problema de maximización escribiendo la siguiente ecuación:

$$\nabla[f(x_2, x_3, x_4) - \lambda g(x_2, x_3, x_4)] = 0 \tag{4.18}$$

A partir de la Ecuación 4.18 obtenemos un sistema de cuatro ecuaciones con cuatro incógnitas; x_2, x_3, x_4 y λ.

$$\frac{\partial}{\partial x_2}\frac{1}{5}\left[(x_{max}-\overline{x})^2+(x_{min}-\overline{x})^2+(x_2-\overline{x})^{\,2}+(x_3-\overline{x})^{\,2}+(x_4-\overline{x})^{\,2}\right]- \lambda\left(x_{max}+x_{min}-5\overline{x}+x_2+x_3+x_4\right)=0 \tag{4.19}$$

$$\frac{\partial}{\partial x_3}\frac{1}{5}\left[(x_{max}-\overline{x})^2+(x_{min}-\overline{x})^2+(x_2-\overline{x})^{\,2}+(x_3-\overline{x})^{\,2}+(x_4-\overline{x})^{\,2}\right]- \lambda\left(x_{max}+x_{min}-5\overline{x}+x_2+x_3+x_4\right)=0 \tag{4.20}$$

$$\frac{\partial}{\partial x_4}\frac{1}{5}\left[(x_{max}-\overline{x})^2+(x_{min}-\overline{x})^2+(x_2-\overline{x})^{\,2}+(x_3-\overline{x})^{\,2}+(x_4-\overline{x})^{\,2}\right]- \lambda\left(x_{max}+x_{min}-5\overline{x}+x_2+x_3+x_4\right)=0 \tag{4.21}$$

$$x_{min}+x_2+x_3+x_4+x_{max}-5\overline{x}=0 \tag{4.22}$$

Simplificando las Ecuaciones 4.19, 4.20 y 4.21 y despejando de la Ecuación 4.22 tenemos:

$$2(x_2-\overline{x})+5\lambda=0 \tag{4.23}$$

$$2(x_3-\overline{x})+5\lambda=0 \tag{4.24}$$

$$2(x_4-\overline{x})+5\lambda=0 \tag{4.25}$$

$$-x_2-x_3-x_4=x_{min}+x_{max}-5\overline{x} \tag{4.26}$$

Matricialmente el sistema de ecuaciones queda planteado como se muestra a continuación:

$$\begin{bmatrix} 2 & 0 & 0 & 5 \\ 0 & 2 & 0 & 5 \\ 0 & 0 & 3 & 5 \\ -1 & -1 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_2 \\ x_3 \\ x_4 \\ \lambda \end{bmatrix} = \begin{bmatrix} 2\overline{x} \\ 2\overline{x} \\ 2\overline{x} \\ min + max - 5\overline{x} \end{bmatrix} \tag{4.27}$$

Resolviendo el sistema de ecuaciones 4.27 obtenemos:

$$x_2 = \frac{(-x_{max} - x_{min} + 5\overline{x})}{3}, \tag{4.28}$$

$$x_3 = \frac{(-x_{max} - x_{min} + 5\overline{x})}{3}, \tag{4.29}$$

$$x_4 = \frac{(-x_{max} - x_{min} + 5\overline{x})}{3} \tag{4.30}$$

$$\lambda = \frac{-2(x_{max} + x_{min} - 2\overline{x})}{15} \tag{4.31}$$

Procedemos a ver si estos valores obtenidos en las Ecuaciones 4.28, 4.29, 4.30 y 4.31 corresponden a valores extremos de la función. Aplicando el determinante de la matriz Hessiana para lo cual calculamos las segundas derivadas parciales.

$$H(x_2, x_3, x_4) = \begin{bmatrix} \frac{\partial^2 f(x_2,x_3,x_4)}{\partial x_2^2} & \frac{\partial^2 f(x_2,x_3,x_4)}{\partial x_2 \partial x_3} & \frac{\partial^2 f(x_2,x_3,x_4)}{\partial x_2 \partial x_4} \\ \frac{\partial^2 f(x_2,x_3,x_4)}{\partial x_3 \partial x_2} & \frac{\partial^2 f(x_2,x_3,x_4)}{\partial x_3^2} & \frac{\partial^2 f(x_2,x_3,x_4)}{\partial x_3 \partial x_4} \\ \frac{\partial^2 f(x_2,x_3,x_4)}{\partial x_4 \partial x_2} & \frac{\partial^2 f(x_2,x_3,x_4)}{\partial x_4 \partial x_3} & \frac{\partial^2 f(x_2,x_3,x_4)}{\partial x_4^2} \end{bmatrix} \tag{4.32}$$

Las primeras derivadas parciales están dadas por las ecuaciones:

$$\frac{\partial f(x_2,x_3,x_4)}{\partial x_2} = \frac{2(x_2 - \overline{x})}{5} \quad \frac{\partial f(x_2,x_3,x_4)}{\partial x_3} = \frac{2(x_3 - \overline{x})}{5} \quad \frac{\partial f(x_2,x_3,x_4)}{\partial x_4} = \frac{2(x_4 - \overline{x})}{5}$$

Por lo que las expresiones de las segundas derivadas parciales son:

$$\frac{\partial^2 f(x_2,x_3,x_4)}{\partial x_2^2} = \frac{2}{5} \quad \frac{\partial^2 f(x_2,x_3,x_4)}{\partial x_2 \partial x_3} = 0 \quad \frac{\partial^2 f(x_2,x_3,x_4)}{\partial x_2 \partial x_4} = 0$$

$$\frac{\partial^2 f(x_2,x_3,x_4)}{\partial x_3 \partial x_2} = 0 \quad \frac{\partial^2 f(x_2,x_3,x_4)}{\partial x_3^2} = \frac{2}{5} \quad \frac{\partial^2 f(x_2,x_3,x_4)}{\partial x_3 \partial x_4} = 0$$

$$\frac{\partial^2 f(x_2,x_3,x_4)}{\partial x_4 \partial x_2} = 0 \quad \frac{\partial^2 f(x_2,x_3,x_4)}{\partial x_4 \partial x_3} = 0 \quad \frac{\partial^2 f(x_2,x_3,x_4)}{\partial x_4^2} = \frac{2}{5}$$

Sustituyendo los valores de las Ecuaciones 4.28, 4.29 y 4.30 en la Ecuación 4.16 y los valores de las derivadas parciales para construir la matriz Hessiana 4.32 y obtenemos:

$$H(x_2, x_3, x_4) = \begin{bmatrix} \frac{2}{5} & 0 & 0 \\ 0 & \frac{2}{5} & 0 \\ 0 & 0 & \frac{2}{5} \end{bmatrix} \tag{4.33}$$

$$\det |H(x_2, x_3, x_4)| = \det \begin{bmatrix} \frac{2}{5} & 0 & 0 \\ 0 & \frac{2}{5} & 0 \\ 0 & 0 & \frac{2}{5} \end{bmatrix} \tag{4.34}$$

$$\det |H(x_2, x_3, x_4)| = \left(\frac{2}{5}\right)^3 = \frac{8}{125} \tag{4.35}$$

Como el determinante $\det |H(x_2, x_3, x_4)| > 0$ se comprueba que los valores pertenecen a un extremo de la función.

Para determinar si esos valores de x_i corresponden a un mínimo o un máximo de la función, se calcula el valor $\frac{\partial^2 f(x_2,x_3,x_4)}{\partial x_2^2}$. Si $\frac{\partial^2 f(x_2,x_3,x_4)}{\partial x_2^2} > 0$ entonces es un mínimo, en el caso contrario es un máximo. En este caso es un mínimo ya que $\frac{\partial^2 f(x_2,x_3,x_4)}{\partial x_2^2} = \frac{2}{5}$.

Usando las Ecuaciones 4.28, 4.29, 4.30 y 4.14, tenemos que el valor mínimo de la varianza se puede calcular a partir de la Ecuación 4.36.

$$\sigma^2_{min} = \frac{2}{15}\left(2x^2_{max} - 5\overline{x}(x_{max} + x_{min}) + x_{max}(x_{min}) + 2x^2_{min} + 5\overline{x}^2\right) \tag{4.36}$$

Ya que los valores anteriores nos proporcionan el mínimo de la función, ahora hay que determinar los valores en los que se alcanza el máximo de la función. Analizando la frontera de la ecuación 4.14 (los valores para los cuales es válida la función), para determinar los valores que maximizan la función. Para esto podemos escribir la Ecuación 4.37:

$$\sigma^2_{max} = \begin{cases} \frac{2}{5}\left(x^2_{max} + 3(x_{max})x_{min} + 6x^2_{min}\right) - 2\overline{x}(x_{max} + 3x_{min}) + 4\overline{x}^2 & 5\overline{x} \leq 2x_{max} + 3x_{min} \\ 4\overline{x}^2 - 4\overline{x}(x_{max} + x_{min}) + \frac{2}{5}\left(3x^2_{max} + 4(x_{max})x_{min} + 3x^2_{min}\right) & 2x_{max} + 3x_{min} < 5\overline{x} < 3x_{max} + 2x_{min} \\ \frac{2}{5}\left(10\ \overline{x}^2 - 5\ \overline{x}(3x_{max} + x_{min}) + 6\ x^2_{max} + 3\ (x_{max})\ x_{min} + x^2_{min}\right) & 3x_{max} + 2x_{min} \leq 5\overline{x} \end{cases} \tag{4.37}$$

Aplicando los resultados de la Ecuaciones 4.36 y 4.37 y simplificando para calcular los límites de confianza en la ecuación 4.4 obtenemos finalmente la Ecuación 4.38:

$$(\overline{x}_A - \overline{x}_B) - t\sqrt{\frac{\sigma_A^2}{5} + \frac{\sigma_B^2}{5}} \leq \mu_A - \mu_B \leq (\overline{x}_A - \overline{x}_B) + t\sqrt{\frac{\sigma_A^2}{5} + \frac{\sigma_B^2}{5}} \tag{4.38}$$

La Ecuación 4.38 será utilizada para el análisis de la calidad de las conexiones por diferentes rutas de un AS. Esta ecuación que se basa en Intervalo de confianza para la diferencia de medias de dos distribuciones normales con varianzas diferentes y desconocidas, [Torre, 2007], donde se sustituye las varianzas de las muestras, por los valores mínimos y máximos calculados previamente. Esto permite, tener una gráfica, donde queda delimitado el rango de valores posibles con una muestra de 5 datos para cada latencia. Esta fórmula, es parte de la propuesta de esta tesis de tener una herramienta para poder comparar de una manera muy práctica y rápida dos latencias diferentes y se utiliza en todas las comparaciones de rutas, donde se puede obtener si una latencia es realmente mejor que otra o si no hay una diferencia significativa (95 % de certeza)

Para cuantificar la latencia, necesitamos el tiempo mínimo, el máximo y el promedio. Con estos tres valores podemos obtener una varianza mínima y máxima con los valores de cada trayecto a comparar y dimensionar. Con 20 o 30 muestras se puede dar más precisión a las medias y a la varianza, y aunque no hay un mínimo del tamaño de las muestras es claro, que, entre más muestras, más tiempo se requiere para obtenerlas, y en ese periodo, las rutas pueden estar cambiando. Es importante para el análisis que cada muestra sea el resultado de la ruta que siguen los paquetes para cada ISP. Estrictamente, el envío solo se puede modificar en el primer salto, y la devolución de los paquetes, podría no seguir la misma ruta. Nuestro método se centra principalmente en conocer la varianza mínima y máxima que tiene cada proveedor, para poder comparar los diferentes caminos. Hay algunas investigaciones sobre el tamaño de muestras muy pequeñas como lo publican en [de Winter, 2013] y [Lee, 1992].

4.2.3. Análisis de la Ruta

En contraste con el método anterior, donde sólo se comparan las latencias numéricas reales, el método de optimizar rutas consiste en acortar los caminos que recorren los paquetes por diferentes sistemas autónomos al analizar los caminos posibles, haciendo estas rutas más eficientes. El router, utilizando BGP

[Elguea and Martinez-Rios, 2017], optimiza las rutas más cortas considerando principalmente el número de sistemas autónomos por los que tiene que pasar el tráfico al utilizar esa ruta.

4.3. Descripción del algoritmo de mejor ruta

En el algoritmo 4.1 se tienen una estructura de datos R_i todas las rutas aprendidas por el Router de los diferentes ISP, cada ruta es identificada por el segmento de red, una lista de los ASs que debe transitar para alcanzar esa red y una dirección del siguiente "Next-hop". Cada R_i es de la forma R_i=[$Segmento_i$, [$AS_{(i,1)}$, $AS_{(i,2)}$, ..., $AS_{(i,n_i)}$],NH_i]. Los vecinos directos (los ASs de los ISP) son todos los $AS_{(i,1)}$ y los n_i son los saltos para alcanzar al segmento de red $Segmento_i$ y los NH_i, son todas las direcciones de los siguientes saltos generalmente asociadas a los $AS_{(i,1)}$ (Los ISP). Esta información es la que se obtiene directamente por BGP, de las tablas de ruteo.

Se define el vector ISP[AS_1, AS_2, ..., AS_j] con los $AS_{(i,1)}$ diferentes de los j-ésimos ISP y definiremos también la dupla $Siguiente_m[AS_m, NH_m]$ para las ocurrencias de los diferentes m siguientes saltos como se ve en el Algoritmo 4.1.

Entonces el objetivo que se busca es crear el subconjunto de Rutas R'_k, si es que existe, de la forma R'_k=[$Segmento_k$, [$AS_{(k,w+1)}$, $AS_{(k,w+2)}$, ..., $AS_{(k,n_i-w)}$], NH'_k] con $w \geq 1$ donde $AS_{(k,w+1)} \in ISP$ y NH'_k, será el otro elemento en $Siguiente_m$ de $AS_{(k,w+1)}$. Es importante notar que la ruta nueva se recorta w saltos, que son los w primero sistemas autónomos que sobran, es decir, se busca si alguno de los ISP a los que se conecta directamente el router, se encuentra en el transcurso de la ruta.

Datos: R_i, las O rutas en el Router; ISP, los vecinos directos
Resultado: $R'_k \subset R_i$
inicio

```
    // Inicializamos R'
    R' ⟵ ∅
    para i ← 1 a O hacer
        Saltos ⟵ n_i
        // Obtenemos los saltos de esa ruta

        // Iniciamos en dos para omitir el AS del ISP original
        para j ← 2 a Saltos hacer
            // Verificamos si AS_(i,j) es elemento de ISP[]
            si AS_(i,j) ∈ ISP[] entonces
                Camino ⟵ [AS_(i,j), AS_(i,j+1), ... AS_(i,n_i)]
                // Al optimizar acortamos las rutas

                // El valor del siguiente salto cambia al cambiar de proveedor, esto se resuelve
                   con el Algoritmo 2
                AgregaA R'[Segmento_i, Camino, NH'_i]
                // Agregamos la nueva ruta al arreglo
            fin
        fin
    fin
fin
```

Algoritmo 4.1: Algoritmo para la selección de las rutas a optimizar.

4.3.1. Comparativo de Rutas

En esta sección veremos ejemplos de ejecuciones del algoritmo desarrollado y cómo obtiene los datos de las respuestas del comando de BGP. También se explica cómo esos datos son utilizados para optimizar las rutas atendiendo.

Los Listados 4.1 y 4.2, muestran el resultado de las mejores rutas para el segmento de red 45.239.78.0/24. En el Listado 4.1 sólo se ven las dos rutas de los ISP de la Universidad Panamericana, donde se distingue que en este caso la mejor ruta es la segunda, la del proveedor con el Sistema Autónomo de 6503 (Axtel) (Listado 4.1, pág. 58, Línea 3 y 9) . En la línea 9, se ve que, aunque el mejor camino es el que inicia en el AS6503, aparece en cuarto lugar el AS22566, que resulta que es vecino directo del AS13679 (Es proveedor de la UP).

El Listado 4.3 y el Listado 4.4, muestran el resultado de las mejores rutas para el segmento de red 2001:0450:2060::/48. En el Listado 4.3 sólo se ven las dos rutas de los ISP de la Universidad Panamericana, donde se distingue que en este caso la mejor ruta es la primera, la del proveedor con el Sistema Autónomo de 6503 (Axtel) (Listado 4.3, pág. 58, Línea 3 y 5) . En la línea 5, se ve que, aunque el mejor camino es el que inicia en el AS6503, aparece como destino el AS3549, que resulta que es vecino directo del AS13679 (Es proveedor de la UP).

En el Listado 4.2 se puede observar en la primera ruta (Listado 4.2, Línea 6 y 7) la agregada por la aplicación desarrollada en esta tesis doctoral, con el parámetro de "localpref" con el valor de 300, y el ID del router con el valor de 1.2.3.4. Esta es la nueva ruta creada y se puede observar que es la ruta definida por el algoritmo de BGP como la mejor de las tres rutas que se tienen en los enlaces.

En el Listado 4.4 se puede observar en la primera ruta (Listado 4.4, Línea 7 y 8) la agregada por la aplicación desarrollada en esta tesis doctoral, con el parámetro de "localpref" con el valor de 300, y el ID del router con el valor de 1.2.3.4. Esta es la nueva ruta creada y se puede observar que es la ruta definida por el algoritmo de BGP como la mejor de las tres rutas que se tienen en los enlaces.

Es importante destacar, que el camino de AS que se agrega con la ruta de mayor prioridad, respeta y reproduce la ruta del proveedor seleccionado para esta ruta. En el caso concreto del router de la Universidad Panamericana, como ejemplo, al no ser un router que anuncie las rutas aprendidas por algún ISP a los demás ISPs, este camino de ASs no se utiliza y no se anuncia a los ISP reales. En el caso de un router general si es

necesario, ya que esos caminos se comparten con los demás routers. Esto se comprobó con una simulación construyendo una red de seis routers con seis ASs diferentes utilizando Quagga como software de ruteo.

Claramente el protocolo BGP selecciona la mejor ruta con el parámetro de menor salto entre sistemas autónomos. Una consecuencia de esto, es que, si se quiere alterar las rutas, la opción para esto es hacer "prepends" es decir, crecer la ruta [Ver Camino más corto en la sección 3.1.1] añadiendo más sistemas autónomos para así evitar usar este camino.

Listado 4.1: Listado de ejecución con dos rutas BGP IPv4.

```
00A-Aplus#sh bgp ipv4 uni 45.239.78.0/24
BGP routing table entry for 45.239.78.0/24, version 16796578
Paths: (2 available, best #2, table default)
Advertised to update-groups:
8
3549 3356 3491 22566 263157
162.97.148.93 from 162.97.148.93 (67.16.168.99)
Origin IGP, metric 3693, localpref 100, valid, external
6503 174 32098 22566 263157
148.245.154.1 from 148.245.154.1 (148.245.204.140)
Origin IGP, localpref 100, valid, external, best
```

Listado 4.2: Listado de ejecución con tres rutas BGP IPv4.

```
00A-Aplus#sh bgp ipv4 uni 45.239.78.0/24
BGP routing table entry for 45.239.78.0/24, version 20632235
Paths: (3 available, best #1, table default)
Not advertised to any peer
22566
201.161.34.97 from 200.57.196.188 (1.2.3.4)
Origin IGP, localpref 300, valid, internal, best
3549 3356 3491 22566 263157
162.97.148.93 from 162.97.148.93 (67.16.168.99)
Origin IGP, metric 3693, localpref 100, valid, external
6503 174 32098 22566 263157
148.245.154.1 from 148.245.154.1 (148.245.204.140)
Origin IGP, localpref 100, valid, external
```

Listado 4.3: Listado de ejecución con dos rutas BGP IPv6.

```
00A-Aplus#sh bgp ipv6 uni 2001:0450:2060::/48
BGP routing table entry for 2001:450:2060::/48, version 12097849
Paths: (2 available, best #1, table default)
Not advertised to any peer
6503 3549
2806:0:0:100::19 (FE80::6AEF:BDFF:FE58:F22) from 2806:0:0:100::19 (148.245.204.
    140)
Origin IGP, localpref 100, valid, external, best
22566 3491 3356 3549
2001:1238::3:1 (FE80::3E8A:B002:2506:6F93) from 2001:1238::3:1 (201.161.6.115)
Origin IGP, localpref 100, valid, external
```

Listado 4.4: Listado de ejecución con tres rutas BGP IPv6.

```
00A-Aplus#sh bgp ipv6 uni 2001:0450:2060::/48
BGP routing table entry for 2001:450:2060::/48, version 12096844
Paths: (3 available, best #1, table default)
Flag: 0x4100
Not advertised to any peer
3549
2001:450:2002:236::9D from 2801:F0:20:F80B:F993:B27B:7236:22C6 (1.2.3.4)
Origin IGP, metric 0, localpref 302, valid, internal, best
6503 3549
2806:0:0:100::19 (FE80::6AEF:BDFF:FE58:F22) from 2806:0:0:100::19 (148.245.204.
    140)
Origin IGP, localpref 100, valid, external
22566 3491 3356 3549
2001:1238::3:1 (FE80::3E8A:B002:2506:6F93) from 2001:1238::3:1 (201.161.6.115)
Origin IGP, localpref 100, valid, external
```

Durante la investigación de esta tesis, se pudo observar que hay rutas que no son óptimas desde el punto de vista del camino a seguir, es decir, existe una mejor ruta que la que está seleccionada en el router. La causa más común de esto se debe a que no se anuncia a todos los ASs en todos los segmentos por igual. Un ejemplo muy común en los ISPs en México, está dado porque dos proveedores que son vecinos entre ellos y además tienen a un proveedor de Internet es USA, al no anunciar todos sus segmentos, pero si al proveedor en USA, provoca que haya rutas entre estos vecinos que parten de México, llegan a USA y nuevamente regresan a México, en lugar de simplemente navegar entre ellos.

La forma de encontrar estas rutas se solucionó en este trabajo doctoral desarrollando el algoritmo que a continuación, explicamos. El proceso es el siguiente:

1. Se determinan los vecinos a los cuales tiene conexión el AS. Estos vecinos son los que aparecen como primer AS en los caminos.
2. Se buscan caminos, donde alguno de los vecinos no aparezca como vecino próximo, pero sí se encuentra en los caminos completos de AS.
3. Se corrigen estos caminos que tienen esa particularidad de no conectar directamente con su vecino si no que lo hace a través de un tercero.

4.3.2. Inyección de Rutas y selección del próximo salto

Para optimizar las rutas, es necesario inyectar la nueva ruta, con los parámetros adecuados para que esa ruta tenga prioridad sobre las que ya tiene el ruteador. Las rutas ya aprendidas de los vecinos, no son eliminadas ni modificadas, por lo que basta terminar la conexión con la aplicación para que regresen

a su funcionamiento previo. La inyección de rutas, también se puede utilizar con fines contrarios a los de optimización propuestos en esta Tesis, por ejemplo, se pueden demorar rutas para su análisis desde el punto de vista de la seguridad o bloquear destinos completos.

En el Algoritmo 4.2 se obtiene la dirección del próximo salto de la ruta optimizada, y de los parámetros que se obtienen de BGP necesarios para corregir las rutas, uno de ellos el parámetro "WEIGHT" no es estándar (es decir puede o no aparecer en una versión de BGP) [Cisco, 2011, Juniper, 2017, Quagga, 2002], por lo que se selecciona el primer parámetro común de todas las implementaciones de BGP que es "LOCAL_PREF". Este parámetro, tiene el valor predeterminado 100 (pero se puede modificar ese valor en la mayoría de las implementaciones de BGP) mientras mayor sea ese parámetro para un destino mayor es la preferencia en BGP de utilizar ese destino.

Datos: $Siguiente_m[AS_m, NH_m]$, las duplas de ISP y los Next-Hop; $AS_{(i,j)}$, El nuevo proveedor seleccionado
Resultado: NH'_i, el Next-Hop para el nuevo ISP
inicio
  `// Este algoritmo encuentra la dirección del Next-Hop asociado al AS de la nueva ruta`
  $NH' \longleftarrow 0$
  `// Inicializamos` NH'
  `// Recorremos toda la colección`
  para $n \leftarrow 1$ **a** m **hacer**
    `// Si encontramos el valor de` $AS_{(i,j)}$
    si $AS_n = AS_{(i,j)}$ **entonces**
      $NH' \longleftarrow AS_n$
      `// Asignamos el valor encontrado del Next-Hop a la ruta optimizada`
    fin
  fin
fin

Algoritmo 4.2: Algoritmo para la selección del Next-Hop de las nuevas rutas optimizadas.

En la aplicación desarrollada para esta tesis, se seleccionó un valor predeterminado para este parámetro de 300, (pero esto puede modificarse en el archivo que contiene el código y se denomina BGPListener.java) ya que es un valor lo suficientemente grande para afectar las rutas que tengan el valor predeterminado, pero también para permitir un espacio de números de preferencia para las rutas que se definan manualmente en el router.

Como ejemplo de lo explicado en el párrafo anterior podemos ver en la Figura 4.2 que se muestra una fracción de las rutas de IPv4 en el router de la UP. En este fragmento de rutas, se muestra en la primera

columna el segmento de red (y con un > la mejor ruta), en la segunda columna el Next-Hop, la métrica, la preferencia local, el peso (sólo disponible en equipos de red de la marca Cisco) y el camino. Por Ejemplo: el segmento de red 0.0.0.0, tiene cuatro rutas y aunque la de menores saltos son la dos y tres con un salto cada una, la primera, (Se originó por iBGP y aunque es el camino más largo con tres sistemas autónomos ASs por transitar en el recorrido), tiene una preferencia local de 300 por lo que es la ruta preferida.

```
BGP table version is 13399483, local router ID is 192.100.201.200
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 *>i 0.0.0.0          201.161.34.97                300      0 6503 3491 22566 i
 *                    201.161.34.97                  0      0 22566 i
 *                    148.245.154.1                         0 6503 i
 *                    189.202.194.198                       0 18734 3257 i
 *   1.0.4.0/24       201.161.34.97                         0 22566 32098 4826 38803 56203 i
 *                    148.245.154.1                         0 6503 174 4826 38803 56203 i
 *>                   189.202.194.198                       0 18734 32098 4826 38803 56203 i
 *>  1.0.4.0/22       189.202.194.198                       0 18734 32098 4826 38803 56203 i
 *                    201.161.34.97                         0 22566 32098 4826 38803 56203 i
 *                    148.245.154.1                         0 6503 174 4826 38803 56203 i
 *>  1.0.5.0/24       189.202.194.198                       0 18734 32098 4826 38803 56203 i
 *                    201.161.34.97                         0 22566 32098 4826 38803 56203 i
 *                    148.245.154.1                         0 6503 174 4826 38803 56203 i
 *>  1.0.6.0/24       189.202.194.198                       0 18734 32098 4826 38803 56203 i
 *                    201.161.34.97                         0 22566 32098 4826 38803 56203 i
 *                    148.245.154.1                         0 6503 174 4826 38803 56203 i
 *   1.0.7.0/24       201.161.34.97                         0 22566 32098 4826 38803 56203 i
 *                    148.245.154.1                         0 6503 174 4826 38803 56203 i
 *>                   189.202.194.198                       0 18734 32098 4826 38803 56203 i
 *   1.0.16.0/24      201.161.34.97                         0 22566 3491 2914 2519 i
 *                    148.245.154.1                         0 6503 174 2914 2519 i
 *>                   189.202.194.198                       0 18734 3257 2914 2519 i
 *   1.0.64.0/18      201.161.34.97                         0 22566 3491 2516 7670 18144 i
 *>                   148.245.154.1                         0 6503 3549 2516 7670 18144 i
 *                    189.202.194.198                       0 18734 174 10026 2519 7670 18144 i
 *   1.0.128.0/24     201.161.34.97                         0 22566 32098 38040 23969 ?
 *                    148.245.154.1                         0 6503 3491 6762 38040 23969 i
```

Figura 4.2: Ejemplo de rutas de IPv4 en BGP.

Uno de los requisitos para la inyección de rutas es el ID del router (este es un conjunto de 4 octetos, números del 1 al 255, separados por puntos, similar a una dirección Ipv4) por lo que se determinó utilizar el ID 1.2.3.4 para la aplicación. Este valor de ID permite darle preferencia a algunos routers, por ejemplo, el 1.1.1.1, pero es lo suficiente menor, para tener preferencia en la selección de las rutas en la mayoría de los casos y así lograr que las rutas sean consideradas por el BPG en el momento de la optimización.

4.3.3. Persistencia de las Rutas y con el mismo número de saltos

En el caso de las rutas nuevas insertadas, si por alguna razón, se interrumpe la conexión entre el sistema y el router, la conectividad de la red no se ve afectada, ya que quedaran las rutas originales de BGP,

pues las rutas originales no son eliminadas, simplemente se agregan rutas con mejores parámetros.

Así mismo, rutas con el mismo número de saltos o sistemas autónomos intermedios, es el protocolo BGP el que decide con base a la configuración, pudiendo seleccionar ambas rutas para balancear, o seguir el orden siguiente: preferir la ruta con el tipo de origen más pequeño, la ruta con el discriminador de salida múltiple más bajo (MED), preferir eBGP sobre rutas iBGP o la ruta con la métrica IGP más baja al siguiente salto BGP.

Capítulo 5

Experimentación y Resultados

En este capítulo se demostrará la relación de la latencia con la distancia del enlace, también, utilizando experimentos factoriales, se prueba, que no existe dependencia con otros factores, como: el protocolo, la saturación del enlace o el tamaño del paquete. Estos argumentos son importantes, pues permiten enfocar los algoritmos en reducir las distancias y así disminuir la latencia.

Los experimentos relacionados con el análisis de la varianza, permiten comparar si existen diferencias entre las medias de las latencias. Si no existe diferencia se puede decir que las latencias utilizando dos rutas son indistintas, pero si existe diferencia, se puede asegurar que la latencia de menor promedio es la mejor ruta. Por ejemplo, entre dos rutas en la misma ciudad, usando diferentes proveedores, las medias pueden ser distintas, pero el análisis de varianza puede indicar que esas diferencias no son significativas, por lo que podemos usar cualquiera de estos dos proveedores. Sin embargo, en el caso de que un proveedor utilice una ruta que salga de la ciudad o incluso del país, esto se evidenciará en el análisis de la varianza como diferencias significativas entre ambos valores de las medias de las latencias. ANOVA se utiliza para validar el método propuesto en esta Tesis que se enfoca en encontrar el rango de la varianza de cada muestra (valor mínimo y valor máximo) y comparar esos intervalos.

Por último en este capítulo, se muestra los diagramas UML de las principales clases que componen la aplicación, así como un resumen de los resultados en la sección 5.4

5.1. Experimento Factorial en redes WAN

Para nuestro trabajo experimental se realizó un experimento factorial para determinar qué parámetros afectan a la latencia y en qué proporción en una red WAN. Se consideraron dos proveedores, ambos con IPv4 e IPV6, dos tamaños de paquetes (100 bytes y 1200 bytes) y utilizamos destinos a dos distancias diferentes (1570 km y 2660 km) y en dos horarios (9:00 a.m. y 1:00 p.m.) para coincidir con los dos

períodos de saturaciones de la red. El experimento siguió las recomendaciones de Breyfogle III, Forrest W [Elguea et al., 2016, Breyfogle III, 1992] para los experimentos factoriales.

Los cinco factores seleccionados se muestran en la Tabla 5.1 y abarcan todas las variables que influyen en la latencia de una conexión en un momento determinado. Adicionalmente se añadieron los proveedores como parámetro, para analizar las diferencias entre cada uno.

Factor	**Nivel Alto (+)**	**Nivel Bajo (-)**	**Valor (ms.)**
Protocolo	IPv6	IPv4	1.5
Saturación	30 % Uso (9:00 am)	15 % Uso (1:00 pm)	-0.125
Tamaño	Paquete de 100 Bytes	Paquete de 1200 Bytes	-1.125
Distancia	Enlace de 2662 Km.	Enlace de 1568 Km.	48.875
Proveedor	AS18734	AS6503	-6.75

Tabla 5.1: Factores seleccionados.

Con los datos de la Tabla 5.1 se procedió a calcular los efectos de los diferentes factores en el valor de la latencia. Los resultados se muestran en la última columna de la Tabla 5.1. Para medir la latencia se utilizó el router principal de la Universidad Panamericana, como se tienen cinco parámetros a experimentar y cada uno de ellos tiene dos opciones, fue necesario realizar $2^5 = 32$ pruebas, cada una de ellas compuesta por 1000 envíos de paquetes en diferentes condiciones.

La perdida de paquetes afecta la confiabilidad de los experimentos. De cada grupo de 1000 paquetes se tomó la latencia promedio. Prácticamente de los 32000 paquetes enviados en las pruebas, sólo se tuvieron 12 paquetes perdidos en 8 de los 32 experimentos, de los cuales, cinco ocurrieron en el período de pruebas de bajo tráfico y siete en el período de tráfico medio por lo que la confiabilidad de los experimentos fue de 99.96 %.

Los resultados de todas las pruebas se muestran en la Tabla 5.2. Esta tabla muestra los resultados de las 32 pruebas y con estos datos se analizarán las relaciones entre ellos y su influencia en la latencia.

Prueba	Protocolo	Paquete	Distancia	Tráfico	Proveedor	Latencia ms
1	+	-	+	-	-	83
2	+	+	+	-	-	82
3	-	-	+	-	-	77
4	-	+	+	-	-	78
5	+	-	-	-	-	21
6	+	+	-	-	-	24
7	-	-	-	-	-	24
8	-	+	-	-	-	24
9	+	-	+	-	+	82
10	+	+	+	-	+	83
11	-	-	+	-	+	75
12	-	+	+	-	+	77
13	+	-	-	-	+	36
14	+	+	-	-	+	38
15	-	-	-	-	+	40
16	-	+	-	-	+	41
17	+	-	+	+	-	83
18	+	+	+	+	-	83
19	-	-	+	+	-	79
20	-	+	+	+	-	80
21	+	-	-	+	-	23
22	+	+	-	+	-	23
23	-	-	-	+	-	24
24	-	+	-	+	-	24
25	+	-	+	+	+	79
26	+	+	+	+	+	85
27	-	-	+	+	+	75
28	-	+	+	+	+	76
29	+	-	-	+	+	36
30	+	+	-	+	+	37
31	-	-	-	+	+	40
32	-	+	-	+	+	40

Tabla 5.2: Resultados de las 32 pruebas del experimento factorial.

Cada renglón representa una variación en las variables del experimento.

5.1.1. Factor Proveedor en redes WAN

Aunque se tienen tres **proveedores** de Internet, la prueba se realizó utilizando a dos de ellos pues con esto es suficiente para evaluar la influencia de cada proveedor en la variación de la latencia. Se verificó, que cada proveedor utilice un trayecto diferente.

5.1.2. Factor Protocolo en redes WAN

Respecto al **protocolo**, el promedio de la latencia es ligeramente mayor en IPv6 por un 2.75 % (54.63 ms vs 56.13 ms) comparado con IPv4, aunque se supone que IPv6 es más eficiente. La diferencia, según muestra la literatura, se debe a que se requieren más equipos en el tramo local de routers, esto es consistente con mediciones realizadas por Cisco [Cisco, 2010a, Cisco, 2010b] para ambos protocolos.

5.1.3. Factor Saturación a en redes WAN

El estudio de la **saturación** del enlace y su relación con la latencia promedio, a las 9:00 a.m. y a la 1:00 p.m., es la que menor diferencia presenta con un valor de 0.12 ms. (55.31 ms a las 9:00 a.m. vs 55.43 a la 1:00 p.m.). Esto indica que el ancho de banda está bien dimensionado. La Figura 5.1 muestra la utilización en horas pico (alrededor de las 11:00 am) donde el uso del enlace no sobrepasa el 35 % y la carga de trabajo a las 9:53 a.m. es de aproximadamente 24 %. La Figura 5.2 muestra la carga de trabajo en la red a las 1:53 p.m. de aproximadamente 18 %. En todos estos períodos se cumple con la recomendación de que los enlaces no se encuentren utilizados más del 80 % del ancho de banda total para adecuarse a los picos de trasmisión.

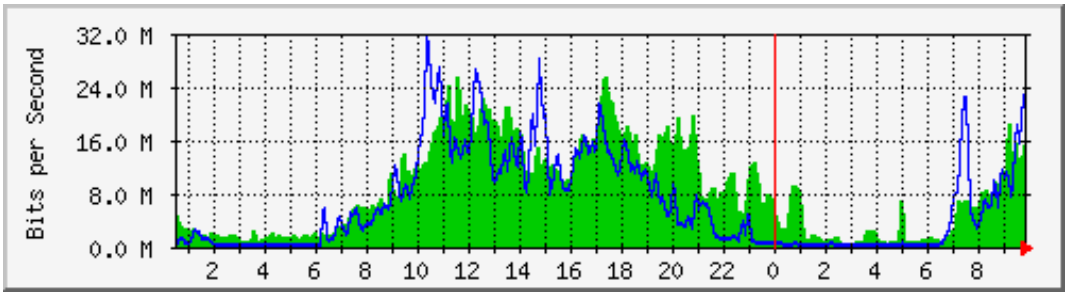

Figura 5.1: Utilización del ancho de banda de AS18734 9:00 am.

Las Figuras 5.1 y 5.2 muestran al mismo proveedor en esos dos horarios. El ancho de banda es de 100 Mb/s de Internet Dedicado, la saturación nunca fue superior al 30 %.

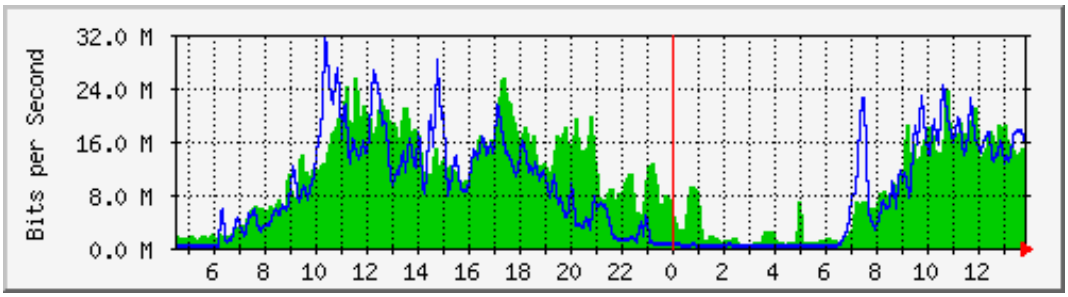

Figura 5.2: Utilización del ancho de banda de AS18734 1:00 pm.

5.1.4. Factor Tamaño del paquete en redes WAN

Con respecto al **tamaño del paquete**, tampoco se nota efecto con respecto al valor de la latencia. Los percentiles de 500 paquetes enviados de los dos tamaños diferentes (100 bytes y 1200 bytes) se muestran en la Figura 5.3 donde la gráfica superior muestra el tamaño para paquetes de 1200 bytes, y la inferior la de paquetes de 100 bytes. Aunque en la latencia mínima (72 vs 73 ms) y en la latencia promedio (73 vs 74 ms) la diferencia fue de sólo 1 ms. En este mismo experimento los resultados mostraron que la diferencia en la latencia máxima fue de 4 ms. Los percentiles de todos los paquetes enviados de los dos tamaños prefijados en el experimento, muestran promedios de latencia similares y que la mayor demora se produce en los paquetes de 1200 bytes (la diferencia fue de 4 ms).

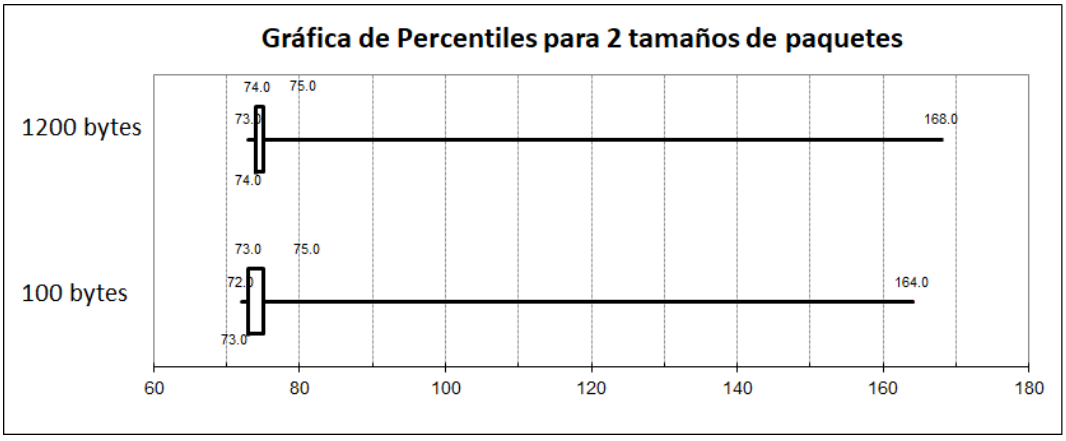

Figura 5.3: Percentiles de dos tamaños de paquetes: 100 bytes y 1200 bytes.

5.1.5. Factor Distancia en redes WAN

Con respecto a la **distancia** entre el origen y el destino, este es el factor que más influye en la latencia por ser el que presenta mayor diferencia en los tiempos de dos conexiones diferentes, 48 ms. Los resultados en este factor son congruentes con la teoría de redes, en el sentido de que la latencia de propagación depende de la longitud del medio y de la velocidad de la señal. Se verificó en las pruebas el trayecto seguido por los paquetes en la red para asegurar que todos siguieran la misma trayectoria.

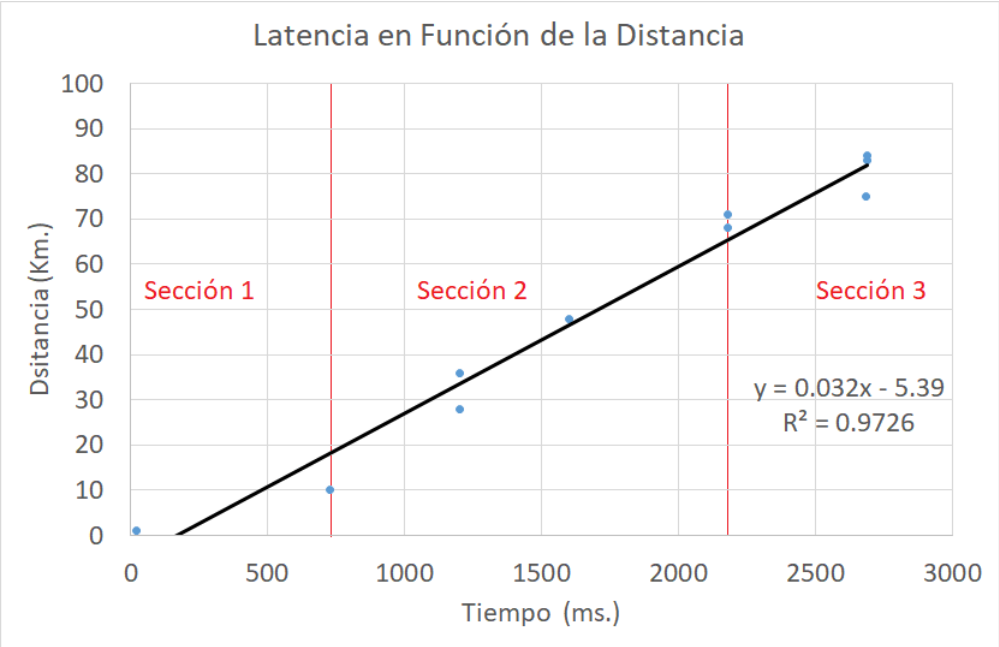

Figura 5.4: Resultados de los experimentos para obtener la latencia en función de la distancia.

En la Figura 5.4 se puede observar el factor de correlación de 0.9726 obtenido al ajustar una línea recta (en color negro) a los valores experimentales obtenidos (en color azul). Este valor de correlación cercano a uno indica una buena descripción de los datos experimentales obtenida con el ajuste de un modelo lineal dado por la ecuación $y = 0.0325x - 5.3979$; en este modelo la "x" representa la distancia en kilómetros y "y" la latencia en milisegundos. Esto demostró claramente que la latencia es una función de la distancia en enlaces WAN.

También se puede notar tres secciones con pendientes muy diferentes y que están marcadas en la Figura 5.4:

1. En la sección 1, el enlace entre la UP y el Proveedor Local (AS18734), la latencia es pequeña, debido a que el enlace es dedicado, es decir, para uso exclusivo de la UP.
2. La segunda sección, muestra el comportamiento más común de las redes, latencias estables para largas distancias, pero variaciones en las conexiones de los diferentes proveedores debido a las latencias propias de cada equipo.
3. En la sección 3 también es un enlace dedicado, por lo que la pendiente de la recta en color azul que se encontró es similar a la pendiente de la primera zona.

La selección del proveedor, prácticamente no influye en la latencia, sobre todo en los últimos segmentos de red, que ya son enlaces internacionales. Las diferencias de las latencias, pueden ser debidas a los switches utilizados [Yang and Cisco, 2012], y aunque son menores que las latencias de los routers y de los enlaces, en determinadas ocasiones deben ser considerados en los análisis (30-50 microsegundos en el Switch comparado con 1 a 2 ms en cada Router).

En los análisis de experimentos factoriales, se consideran dos tipos de interacciones: la fuerte y la débil. Se tiene interacción Fuerte entre dos factores cuando sus gráficas se intersectan como se muestra en las Figuras 5.5 y 5.6.

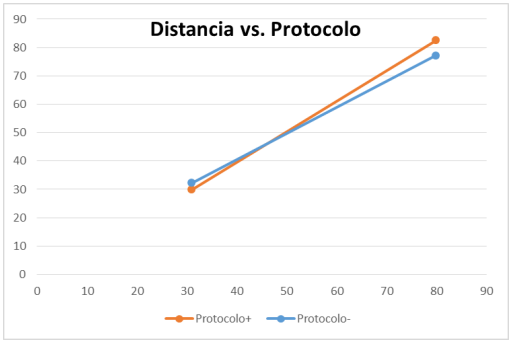

Figura 5.5: Interacción fuerte de los factores Distancia y Protocolo.

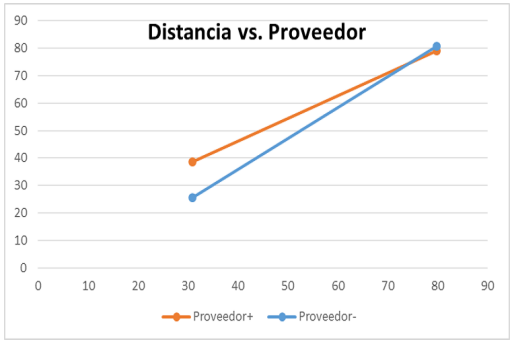

Figura 5.6: Interacción fuerte de los factores Distancia y Proveedor.

Se tiene interacción débil entre dos factores cuando sus gráficas no se intersectan en la sección de los valores de experimentación, pero debido a sus diferentes pendientes, se intersectan en algún punto fuera de

los límites de los valores experimentados como se muestra en las Figuras 5.7 y 5.8.

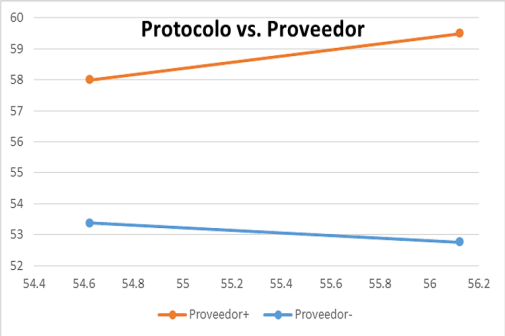

Figura 5.7: Interacción débil de los factores Protocolo y Proveedor.

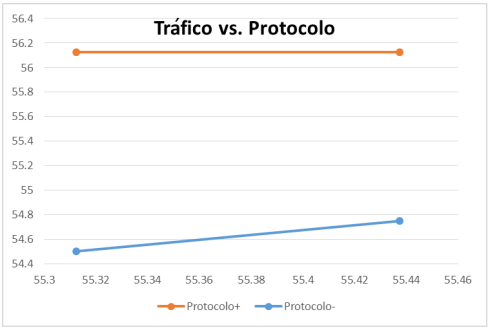

Figura 5.8: Interacción débil de los factores Tráfico y Protocolo.

Para la obtención de estos datos se elaboró una aplicación en Java (Anexo: A.17), que cada cinco minutos se conecta al router de frontera, realiza una cantidad de 50 ping al destino (mensajes enviados entre dos equipos y del cual se obtiene el tiempo de respuesta) con un tamaño de datos de 1000 bytes y especifica las interfaces de salida de cada proveedor para asegurar que se utilice la dirección y la ruta de cada uno de los proveedores. Los resultados de cada prueba (% de paquetes recibidos, tiempo mínimo, promedio y tiempo máximo), se van almacenando en un archivo de texto para posteriormente ser procesados. En el Listado 5.1 se muestra en las líneas de código 61, 63 y 65, cómo se generan las instrucciones de ICMP para las diferentes interfaces del router, y así poder medir la latencia de cada ISP.

Listado 5.1: Latencia.java

```
60 readUntil(promptCompleteEna);
61 write("ping "+destino+" size "+tamanio+" source GigabitEthernet0/0 repeat "+Cant);
62 AS6503=readUntil(promptCompleteP);
63 write("ping "+destino+" size "+tamanio+" source GigabitEthernet0/1.8 repeat "+Cant);
64 AS22566=readUntil(promptCompleteP);
65 write("ping "+destino+" size "+tamanio+" source GigabitEthernet0/1.10 repeat "+Cant);
66 AS18734=readUntil(promptCompleteP);
```

Respecto a la latencia mínima, promedio y máxima, se obtuvieron datos hacia el mismo sitio en Internet: www.google.com. Esta latencia se midió mediante el envío de 50 paquetes de ICMP cada cinco minutos desde las 10:30 hasta las 13:25 del mismo día.

5.1.6. Análisis del Experimento Factorial

Se demuestra en este experimento que la distancia recorrida por el paquete es el principal factor que afecta a la latencia y que los demás factores: tamaño del paquete, protocolo, saturación, proveedor, etc; no influyen en el tiempo de propagación. Esto permite enfocar los algoritmos con el objetivo de disminuir la distancia recorrida por la señal en la red.

Esta relación directa entre distancia del origen al destino y la latencia, es más importante aún en servicios que utilizan la nube, como lo menciona [Gessert et al., 2020] y en sistemas distribuidos como en [Myers and Hughes, 2020, Sminesh et al., 2020].

5.2. Experimentos de análisis de varianza (ANOVA) y comparación con algoritmo propuesto

En esta sección se compara la latencia utilizando análisis de la varianza o Anova. Este método consiste en comparar la varianza de las medias de las rutas iniciales con la varianza de las medias de las rutas optimizadas, para determinar si las latencias son similares o proceden de rutas con características diferentes. Para este proceso aleatoriamente el algoritmo decidía, si medir primero la latencia de la mejor ruta o de la ruta sin optimizar.

Se presentan dos análisis completos, uno realizado sobre IPv4 y otro sobre IPv6. En estos experimentos se calculan las varianzas de 3600 muestreos realizados en tiempos aleatorios entre 0.5 segundos y 1.5 segundos (en promedio se realizó un muestreo cada 1 segundo durante 5 minutos). Con estos muestreos de medición de latencia en la red se construyeron las gráficas de distribución y se compararon entre sí utilizando el método propuesto en este trabajo y que fue explicado en el Capítulo 4.2.2 utilizando las Ecuación 4.38.

Las condiciones previas para aplicar Anova correctamente [Tomarken and Serlin, 1986] es que se tengan datos de poblaciones que sigan una distribución normal con varianzas iguales, aunque también suele ser efectivo con pequeñas variaciones si no se cumplen dichos supuestos. Estudios previos modelan la distribución de la latencia con una distribución Gamma, LogNormal o Exponencial [Karakaş, 2003], adicionalmente, se pueden normalizar utilizando logaritmos de los tiempos.

Para este análisis, se han separado las rutas en dos categorías. La primera categoría está conformada por rutas donde el destino se encuentra relativamente cerca del origen (por ejemplo, en redes en el mismo país

o en la misma ciudad). En este grupo las latencias se deben encontrar en los rangos de 3 a 15 milisegundos en las mejores rutas. La segunda categoría está compuesta por rutas donde para alcanzar el destino, las latencias alcanzan valores del orden de 50 o más milisegundos. Como referencia a los datos anteriores señalamos que la latencia promedio medida de la Ciudad de México a Singapur es de 130 ms (la dirección de acceso utilizada fue pch.sgzones.sg 204.61.216.57 2001:500:14:6057:ad::1) y a Indonesia alcanza los 160 ms (la dirección de acceso utilizada fue e.dns.id 103.19.177.177 2001:df5:4000:4::4)

5.2.1. Experimentos Anova en redes cercanas

Experimentos de análisis de latencias con el protocolo IPv4

La Tabla 5.3 muestra los principales indicadores para este análisis. Las unidades están expresadas en milisegundos. Nos enfocaremos en el promedio y la varianza. En dicho conjunto de datos se aprecia que el promedio es menor en la ruta más corta, pero la varianza es mayor, lo que implica en este caso, que las variaciones debido a factores que no son la distancia, suelen afectar de forma más importante.

Ruta más Corta		**Ruta más Larga**	
Media	**4.9**	**Media**	**33.24**
Error típico	0.1	Error típico	0.06
Mediana	3.0	Mediana	33.00
Moda	3.0	Moda	32.00
Desviación estándar	**6.2**	**Desviación estándar**	**3.71**
Varianza de la muestra	38.8	Varianza de la muestra	13.73
Curtosis	25.4	Curtosis	188.57
Coeficiente de asimetría	4.7	Coeficiente de asimetría	11.18
Rango	71.0	Rango	85.00
Mínimo	2.0	Mínimo	32.00
Máximo	73.0	Máximo	117.00
Suma	17827	Suma	119670
Cuenta	3600	Cuenta	3600

Tabla 5.3: Resumen estadístico de mediciones de la latencia sobre IPv4 en redes cercanas.

La Tabla 5.4, muestra los resultados del análisis de Anova. El valor de p en la última columna de la tabla, en este caso de cero, indica que la probabilidad estadística bajo la hipótesis nula es menor al nivel de significancia, que es de 0.05. Esto implica que los dos conjuntos de datos no se relacionan, por lo que en este experimento se puede concluir que el conjunto de menor promedio se corresponde con la mejor ruta.

La Figura 5.9 muestra los resultados de las mediciones de latencia en IPv4, utilizando la metodología propuesta en este trabajo doctoral. En la Figura 5.9 la línea continua (en color rojo una y azul la otra respectivamente) muestra los datos obtenidos para la latencia máxima y la línea punteada (en color amarillo

Grupos	Cuenta	Suma	Promedio	Varianza
Ruta más Corta	3600	17827	4.9	38.8
Ruta más Larga	3600	119670	33.24	13.73

ANÁLISIS DE VARIANZA

Origen de la varianza	Suma cuadrados	Grados de libertad	Promedio cuadrados	F	p
Entre grupos	1440555	1	1440555	54800	0
Dentro de los grupos	189216	7198	26.29		
Total	1629771	7199			

Tabla 5.4: Resultados del análisis Anova de latencias en IPv4 en redes cercanas.

y verde respectivamente) los datos para mediciones de la mínima latencia. Estas figuras permiten comparar visualmente las variaciones en las latencias, (Gráficas mas anchas, significan más variación) y ubicar los promedios de las latencias, por lo que si las gráficas están separadas, significa que si hay diferencias significativas, y en el caso de traslaparse, suele ser indistinto que ruta es mejor. La Tabla 5.5 muestra el resultado de

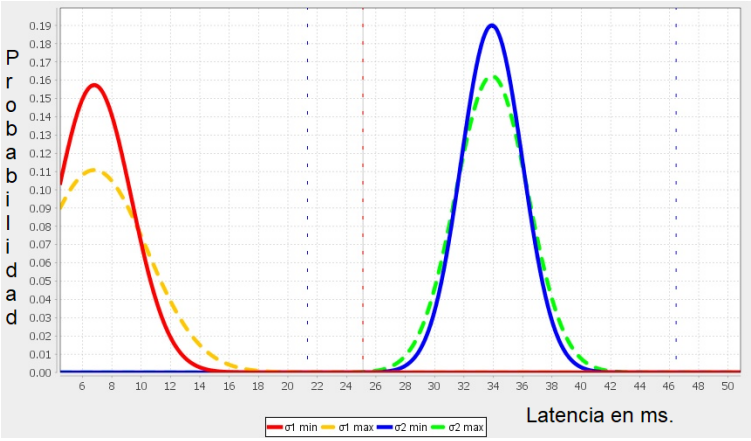

Figura 5.9: Latencias en IPv4 para determinar los límites de confianza en redes cercanas.

los experimentos realizados con el método propuesto en esta tesis. Los resultados de los límites de confianza con varianza mínima y máxima, indican que sí hay diferencia entre los dos conjuntos de datos, por lo que la mejor ruta es la de menor promedio de latencia. En este experimento, los grados de libertad obtenidos para ambas rutas fueron 7.

Experimentos de análisis de latencias con el protocolo IPv6

La Tabla 5.6 muestra los principales indicadores estadísticos obtenidos de los experimentos realizados de análisis de latencias. Nos enfocaremos en el promedio y la varianza. En este conjunto de datos, claramente

	x_{min}	$\overline{x}$	x_{max}	σ^2_{min}	σ^2_{max}	ν	t-Student
Ruta Corta	3.0	6.8	11.0	6.43	12.96	7 (α=95 %)	2.365 (ν=7)
Ruta Larga	73	76	83	12.67	16		
Varianza Mínima						Varianza Máxima	
$-30.58 < \overline{x}_A - \overline{x}_B < -23.50$						$-31.71 < \overline{x}_A - \overline{x}_B < -22.49$	

Tabla 5.5: Límites de confianza para dos rutas diferentes en IPv4 en redes cercanas.

el promedio de latencias y su varianza es menor en la ruta más corta, lo que implica que las variaciones de latencia debido a otros factores, excepto la distancia del enlace, son despreciables.

Ruta más Larga		**Ruta más Corta**	
Media	**38.40**	**Media**	**3.45**
Error típico	0.06	Error típico	0.02
Mediana	38	Mediana	3
Moda	38	Moda	3
Desviación estándar	**3.46**	**Desviación estándar**	**0.92**
Varianza de la muestra	12.02	Varianza de la muestra	0.86
Curtosis	1320.57	Curtosis	116.63
Coeficiente de asimetría	29.74	Coeficiente de asimetría	8.22
Rango	163	Rango	18
Mínimo	37	Mínimo	3
Máximo	200	Máximo	21
Suma	138250	Suma	12422
Cuenta	3600	Cuenta	3600

Tabla 5.6: Resumen estadístico de mediciones de la latencia sobre IPv6 en redes cercanas.

La Tabla 5.7, muestra los resultados del análisis de Anova. El valor de p, en este caso de cero, indica que la probabilidad estadística bajo la hipótesis nula es menor al nivel de significación, que es de 0.05. Esto implica que los 2 sets de datos no se relacionan, por lo que en este experimento se puede concluir que el de menor promedio es también la mejor ruta.

Grupos	Cuenta	Suma	Promedio	Varianza
Ruta más Larga	3600	138250	38.40	12.02
Ruta más Corta	3600	12422	3.45	0.86

ANÁLISIS DE VARIANZA					
Origen de la varianza	Suma cuadrados	Grados de libertad	Promedio cuadrados	F	**P**
Entre grupos	2198984	1	2198984	341663	**0**
Dentro de los grupos	46327	7198	6.44		
Total	2245311	7199			

Tabla 5.7: Resultados del análisis Anova de latencias en IPv6 en redes cercanas.

La Figura 5.10 muestra los resultados de las mediciones de latencia en IPv6, utilizando la metodología

propuesta en este trabajo doctoral. La línea, en color rojo para la ruta no optimizada y en color azul para la ruta optimizada, muestran las diferencias en los datos obtenidos para la latencia. En esta figura, las varianzas mínimas y máximas de cada ruta son las mismas, por lo que no se aprecian las 2 gráficas por cada latencia, sin embargo, la ruta que tiene menor promedio también presenta menor variación, lo que demuestra que es preferible esa ruta.

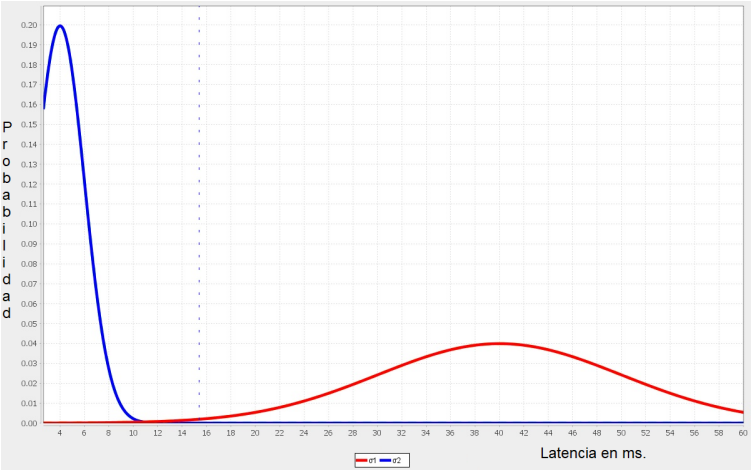

Figura 5.10: Latencias en IPv6 para determinar los límites de confianza en redes cercanas.

La Tabla 5.8 muestra el resultado del algoritmo propuesto. En este caso, el resultado de los límites de confianza con varianza mínima y máxima, indican que sí hay diferencia entre los dos conjuntos de datos, por lo que la mejor ruta es la de menor promedio de latencia. En este experimento, los grados de libertad para ambas rutas es diferente, siendo de 6 para la varianza mínima y de 5 para la máxima.

Tabla 5.8: Límites de confianza para dos rutas diferentes en IPv6.

	x_{min}	$\overline{x}$	x_{max}	σ^2_{min}	σ^2_{max}	ν	t-Student
Ruta Corta	3.0	3.2	4.0	0.16	0.16	6 (α=95 %)	2.571 (ν=6)
Ruta Larga	32	33	34	0.4	0.8	5 (α=95 %)	2.365 (ν=5)
Varianza Mínima						Varianza Máxima	
$-30.619 < \overline{x}_A - \overline{x}_B < -28.868$						$-30.926 < \overline{x}_A - \overline{x}_B < -28.674$	

Tabla 5.9: Límites de confianza para dos rutas diferentes en IPv6 en redes cercanas.

Después de analizar la latencia en los dos protocolos, IPv4 e IPv6, desde la Universidad Panamericana a Netflix, se obtuvieron los datos de la Tabla 5.10, en esta tabla se muestran los límites de confianza para la varianza mínima, usando la ecuación 4.36. Los grados de libertad ν son 4, y la estadística t se busca en

tablas con α=0.025 %.

	x_{min}	$\overline{x}$	x_{max}	σ^2_{min}	ν	t-Student	Límites con varianza Mínima
IPv4	75.0	75.6	77.0	0.507	4 (α=95 %)	2.776 (ν=4)	$-5.268 < \overline{x}_A - \overline{x}_B < 3.886$
IPv6	73.0	76.6	83.0	11.307			

Tabla 5.10: Varianza Mínima de ambos protocolos IPv4 e IPv6 en redes cercanas.

En la Tabla 5.11 se muestra los límites de confianza para la varianza mínima, usando la ecuación 4.37. Los grados de libertad ν también son 4, y como existe cambio de signo en los extremos del límite de confianza, se puede asegurar **con un 95 % de certeza que no existe diferencia significativa entre los dos protocolos** ya que hay cambio de signo entre los límites de la varianza, lo que implica, que las gráficas se entrelazan como se muestra en la Figura 5.11.

	x_{min}	$\overline{x}$	x_{max}	σ^2_{max}	ν	t-Student	Límites con varianza Máxima
IPv4	75.0	75.6	77.0	0.640	4 (α=95 %)	2.776 (ν=4)	$-6.619 < \overline{x}_A - \overline{x}_B < 4.619$
IPv6	73.0	76.6	83.0	19.840			

Tabla 5.11: Varianza Máxima de ambos protocolos IPv4 e IPv6 en redes cercanas.

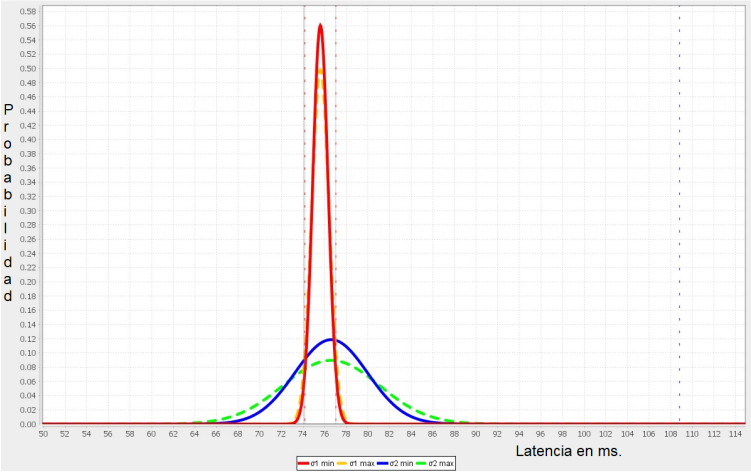

Figura 5.11: Comparativo de latencias de los protocolos IPv4 Vs. IPv6 hacia Netflix.

5.2.2. Experimentos Anova en redes lejanas

Experimentos de análisis de latencias con el protocolo IPv4

La Tabla 5.12 muestra los principales indicadores estadísticos de los experimentos realizados en redes lejanas. En este análisis, nos enfocaremos en el promedio y la varianza de las latencias. En este conjunto de datos, los promedios son muy similares en ambas rutas.

Ruta más Larga		**Ruta más Corta**	
Media	**39.0**	**Media**	**35.10**
Error típico	0.1	Error típico	0.03
Mediana	37	Mediana	35
Moda	37	Moda	35
Desviación estándar	**6.1**	**Desviación estándar**	**1.61**
Varianza de la muestra	36.8	Varianza de la muestra	2.59
Curtosis	19.3	Curtosis	622.65
Coeficiente de asimetría	3.5	Coeficiente de asimetría	10.61
Rango	89.0	Rango	92.00
Mínimo	0.0	Mínimo	0.00
Máximo	89.0	Máximo	92.00
Suma	140537	Suma	126358
Cuenta	3600	Cuenta	3600

Tabla 5.12: Resumen estadístico de la latencia sobre IPv4 en redes lejanas.

La Tabla 5.13, muestra los resultados del análisis de Anova. El valor de la probabilidad, en la última columna de la Tabla 5.13, es cero, lo que indica que la probabilidad estadística bajo la hipótesis nula es menor al nivel de significancia, que es de 0.05. Esto implica que los dos sets de datos no se relacionan, por lo que en este experimento se puede concluir que el resultado de menor promedio en la latencia corresponde a la mejor ruta.

Grupos	Cuenta	Suma	Promedio	Varianza
Ruta más Larga	3600	140537	39.0	36.76737605
Ruta más Corta	3600	126358	35.10	2.59

ANÁLISIS DE VARIANZA					
Origen de la varianza	Suma cuadrados	Grados de libertad	Promedio cuadrados	F	**p**
Entre grupos	27922	1	27922	1418.7	**0**
Dentro de los grupos	141666	7198	19.68		
Total	169588	7199			

Tabla 5.13: Anova de latencias en IPv4 redes lejanas.

La Figura 5.12 muestra los resultados de mediciones de latencia en el protocolo IPv4 en redes lejanas. La línea continua (en colores rojo y azul) muestra las latencias máximas y la línea punteada (en colores

amarillo y verde) las latencias mínimas. (La azul y verde son de la ruta más corta y la roja y amarillo de la ruta más larga)

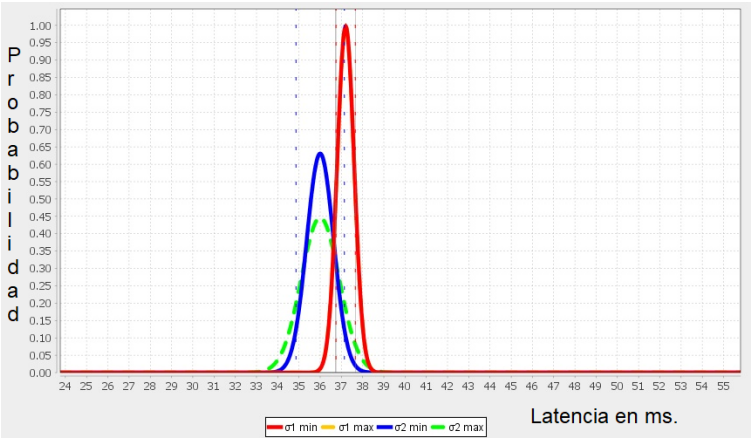

Figura 5.12: Límites de confianza latencias en IPv4 en redes lejanas.

Experimentos de análisis de latencias con el protocolo IPv6

La Tabla 5.14 muestra los principales indicadores estadísticos obtenidos de los experimentos realizados de análisis de latencias. Nos enfocaremos en el promedio y la varianza. En este conjunto de datos, en el promedio de latencias no existe tanta diferencia como en redes cercanas, pero su varianza es menor en la ruta más corta, lo que puede deberse a que en redes lejanas la optimización de rutas, algunas veces no es tan notoria.

Ruta más Larga		**Ruta más Corta**	
Media	**29.5**	**Media**	**29.2**
Error típico	0.0	Error típico	0.0
Mediana	29.4	Mediana	29.2
Moda	29.4	Moda	29.2
Desviación estándar	**0.5**	**Desviación estándar**	**0.2**
Varianza de la muestra	0.2	Varianza de la muestra	0.0
Curtosis	41.3	Curtosis	40.8
Coeficiente de asimetría	6.0	Coeficiente de asimetría	-1.3
Rango	5.5	Rango	4.1
Mínimo	27.7	Mínimo	27.3
Máximo	33.2	Máximo	31.4
Suma	106232	Suma	104966
Cuenta	3600	Cuenta	3600

Tabla 5.14: Resumen estadístico de la latencia sobre IPv6 en redes lejanas.

La Tabla 5.15, muestra los resultados del análisis de Anova. El valor de la probabilidad, en la última columna de la Tabla 5.15, es cero, lo que indica que la probabilidad estadística bajo la hipótesis nula es menor al nivel de significancia, que es de 0.05. Esto implica que los dos sets de datos no se relacionan, por

lo que en este experimento se puede concluir que el resultado de menor promedio en la latencia corresponde a la mejor ruta.

Grupos	Cuenta	Suma	Promedio	Varianza
Ruta más Larga	3600	106232	29.5	0.21
Ruta más Corta	3600	104966	29.1	0.041

ANÁLISIS DE VARIANZA					
Origen de la varianza	Suma cuadrados	Grados de libertad	Promedio cuadrados	F	**p**
Entre grupos	222.65	1	222.65	1753.99	**0**
Dentro de los grupos	913.70	7198	0.127		
Total	1136.34	7199			

Tabla 5.15: Anova de latencias en IPv6 redes lejanas.

La Figura 5.13 muestra los resultados de mediciones de latencia en el protocolo IPv6 en redes lejanas. La línea continua (en colores rojo y azul) muestra las latencias máximas y la línea punteada (en colores amarillo y verde) las latencias mínimas (La azul y verde son de la ruta más corta y la roja y amarillo de la ruta más larga).

Es importante destacar, en este caso, que el algoritmo propuesto en esta tesis indica que no hay diferencia práctica en la latencia al usar la ruta corta sobre la ruta larga, contrario al análisis ANOVA que concluye que si son independientes; por lo que la de mejor promedio si es la más corta. Además, como se muestra en la Figura 5.13, se obtiene un beneficio en la varianza, que implica que la ruta corta es más estable que la ruta larga, y en este caso, podría ser adecuado optimizarla. Adicionalmente a esto, el algoritmo de esta tesis tiene otras ventajas respecto a la velocidad de ejecución y rapidez para identificar la mejor ruta.

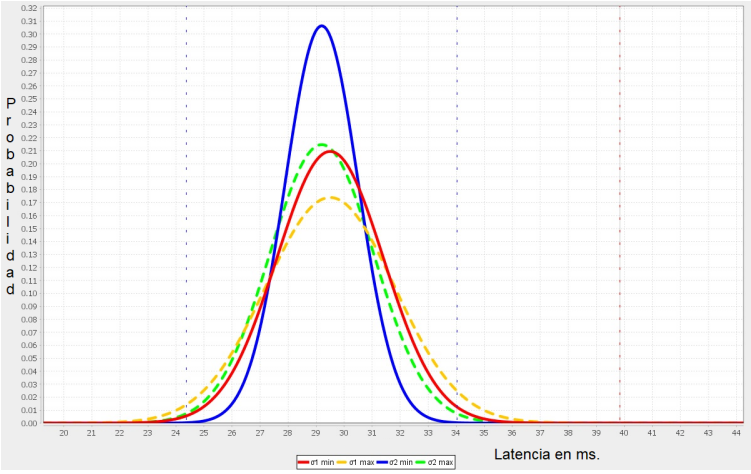

Figura 5.13: Límites de confianza latencias en IPv6 en redes lejanas.

5.3. Desarrollo de la aplicación para la optimización de rutas sobre BGP

Se desarrolló con base en los estándares de BGP versión 4 y un proyecto de OpenSource de BGP de 1999 [Jimenez, 1999] una programa escrito enteramente en Java cuya finalidad es tener una herramienta que permita al usuario visualizar, procesar y obtener información de las rutas de BGP.

Estos estándares permiten al usuario visualizar las rutas de BGP, adicional en nuestro proyecto agregamos funciones para mostrar caminos que pasan por determinados países, rutas más cortas y rutas más largas, así como la distribución de los “Next-Hop”, y sobre todo se implementaron los algoritmos para determinar las rutas a optimizar y los procesos para actualizar dichas rutas.

Con la herramienta se muestran parámetros como el path, el “Next-Hop”, y se puede analizar los parámetros de las conexiones y de los paquetes que contienen las actualizaciones de rutas. También se le añadió la posibilidad de inyectar rutas a los vecinos de BGP, de igual forma, la creación de reportes y análisis más detallados.

La aplicación en Java consiste en una clase principal en donde se definen los parámetros de BGP, como el router id y el Sistema Autónomo, y ejecuta el proceso que atiende las peticiones de red en el puerto 179 de TCP que es el predeterminado de BGP. Por cada petición recibida, se tiene una conexión. A cada conexión, se le envían periódicamente, cada 90 segundos, los mensajes adecuados para mantener la sesión abierta, los mensajes de tipo KeepAlive (Ver Mensajes de BGP en la sección 3.3).

En cada conexión, se almacenan los datos propios de esa sesión de BGP, como son los parámetros

definidos en las sesiones, ver Tabla 3.4, así como también los atributos de las actualizaciones (Tabla 3.5). En el código desarrollado, se implementan los sistemas autónomos de 32 bits, el soporte de IPv6 y la programación de los algoritmos de esta tesis, necesarios para determinar las rutas que se pueden optimizar y opcionalmente inyectar las rutas ya optimizadas a los vecinos de BGP.

Adicional a las funciones de BGP, nuestro código implemento procesos como:

- Análisis de rutas.
- Determinación del nombre y país de cada sistema autónomo.
- Módulos para el procesamiento de las rutas.
- Exportar la información en varios formatos como Excel, PDF y Mapas Temáticos.
- Herramientas para análisis estadístico.

A continuación se muestran las clases principales desarrolladas para la aplicación. El código completo puede ser consultado en el Anexo A.1. En la Figura 5.24 se muestran las relaciones entres las clases principales de la aplicación, todas ellas relacionadas con el protocolo de BGP o con el direccionamiento de red. El modelado en UML de la aplicación, permite visualizar la estructura de cada case, las instancias (los objetos en Java) y sus métodos definidos y es el estándar mas utilizado, [Bruegge and Dutoit, 2009] para modelado de programación orientada a objetos.

La aplicación desarrollada está compuesta por las siguientes clases de Java:

5.3.1. BGPHeader.java

Este archivo se encuentra en el Anexo A.1 y su diagrama se puede ver en la Figura 5.14, contiene la definición que implementa los parámetros necesarios para la conexión de BGP, así como las siguientes contantes que se utilizan en los estados de las conexiones:

- public static byte TYPE_OPEN = 1;
- public static byte TYPE_UPDATE = 2;
- public static byte TYPE_NOTIFICATION = 3;
- public static byte TYPE_KEEPALIVE = 4;

5.3.2. BGPMain

Esta es la clase principal de la aplicación, su diagrama UML es la Figura 5.15, y este proceso es donde se reciben los datos de las conexiones en el puerto 179. Aquí se encuentra el código para que el usuario

pueda interactuar con la aplicación y solicitarle que busque las rutas a optimizar o inyectar las nuevas rutas. También esta clase permite al usuario exportar las rutas como archivo de texto, en formato ARFF para procesarlo en Weka y se pueden generar reportes en LaTeX. El código se encuentra en el Anexo A.2

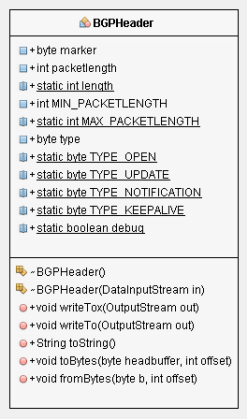

Figura 5.14: UML de BGPHeader

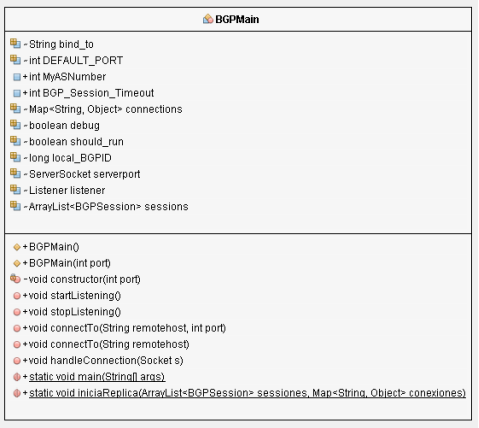

Figura 5.15: UML de BGPMain

5.3.3. BGPNotificationPacket

La Figura 5.16 muestra la clase en Java, donde se definen las notificaciones, los mensajes de error y las funciones de comunicación con otros Routers. Aquí también se definen los diferentes estados que puede tener una conexión de BGP. El código se encuentra en el Anexo A.3.

5.3.4. BGPOpenPacket

En este archivo en Java, está la definición de la clase que se encarga de la estructura del paquete que se envía o se recibe usando BGP, por ejemplo, el ID del router y la versión, y los parámetros que acepta el vecino que se conecta. En el Anexo A.4 se encuentra el código y su diagrama UML se muestra en la Figura 5.17.

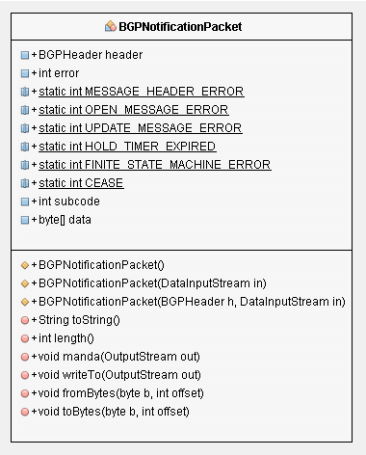

Figura 5.16: UML de BGPNotificationPacket

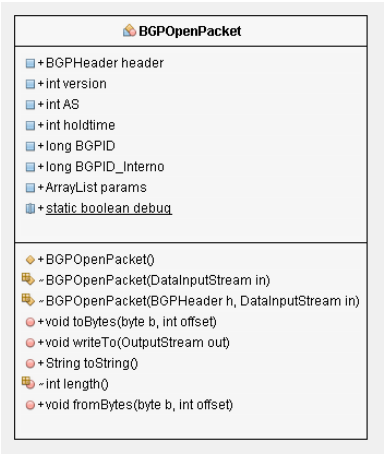

Figura 5.17: UML de BGPOpenPacket

5.3.5. BGPOpenParameter

Las actualizaciones de las rutas de BGP, pueden contener muchos parámetros, aquí están definidos los más usados según los estándares en lo que se basa BGP, por ejemplo, el multiprotocolo que se requiere para usar IPv6 y el soporte para los números de los sistemas autónomos de 32 bits. En sesiones experimentales con grandes redes, se han encontrado incluso parámetros ya marcados como obsoletos o creados para fines muy específicos. La Figura 5.18 muestra sus métodos y atributos. El código se encuentra en el Anexo A.5.

5.3.6. BGPPathAttribute

La Figura 5.19 es el diagrama UML de la clase que se encarga de los paquetes de red y de las actualizaciones de rutas, una parte importante es la que se encarga de la información relacionada con el Path. Este archivo implementa las constantes para la correcta identificación de las partes de esos paquetes. El código se encuentra en el Anexo A.6.

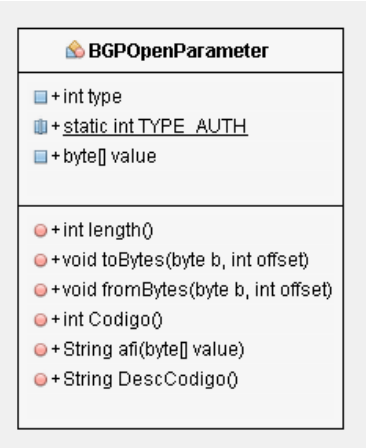

Figura 5.18: UML de BGPOpenParameter

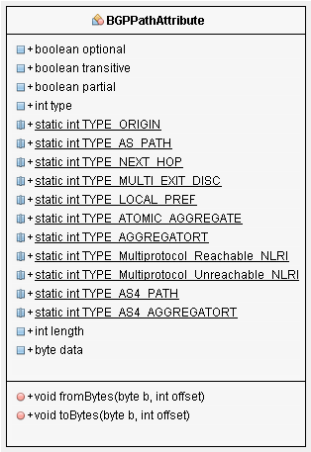

Figura 5.19: UML de BGPPathAttribute

5.3.7. BGPSendUpdatePacket

Cuando se requiere enviar una actualización de red, este código, arma la estructura del paquete que se tiene que enviar. Convierte los datos a binario, para crear la actualización de acuerdo a los estándares de BGP. El código se encuentra en el Anexo A.7 (pág. 148) y su diagrama es la Figura 5.20.

5.3.8. BGPStatus

Esta clase se encarga de almacenar el estado de cada conexión, se lleva un registro de cuántos anuncios de actualizaciones, cuántos segmentos se han dado de alta y cuántos de baja., así como el tiempo que lleva la conexión activa. Este reporte se genera, cada vez que se escribe “st” en la línea de instrucciones. El código se encuentra en el Anexo A.8 y su diagrama es la Figura 5.21.

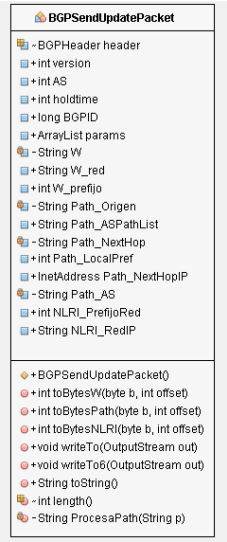

Figura 5.20: UML de BGPSendUpdatePacket

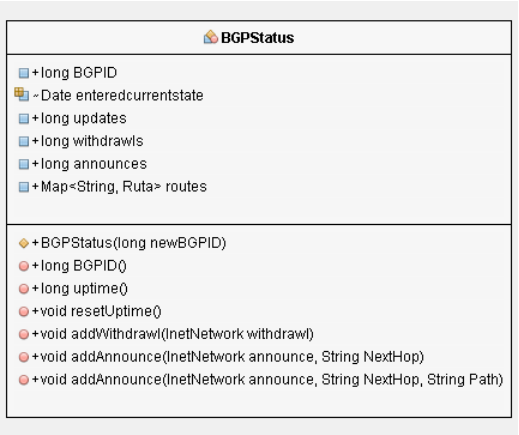

Figura 5.21: UML de BGPStatus

5.3.9. BGPUpdatePacket

Este módulo se encarga de la parte en concreto de la recepción de las actualizaciones. Es la parte equivalente a BGPSendUpdatePaccket pero para el procesamiento de la actualización recibida. La Figura 5.22 muestra su digrama UML y el código se encuentra en el Anexo A.9

5.3.10. InetNetwork

El la Figura 5.23 se encuentra el UML de las herramientas que se encargan del agrupamiento y codificación de las direcciones de red a cadenas de caracteres para su procesamiento en la aplicación. Por ejemplo, almacena segmentos de red, como una dirección más un prefijo del tamaño del segmento. El código se encuentra en el Anexo A.10.

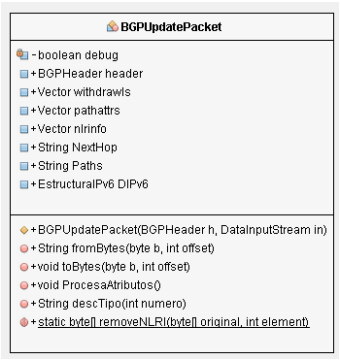

Figura 5.22: UML de BGPUpdatePacket

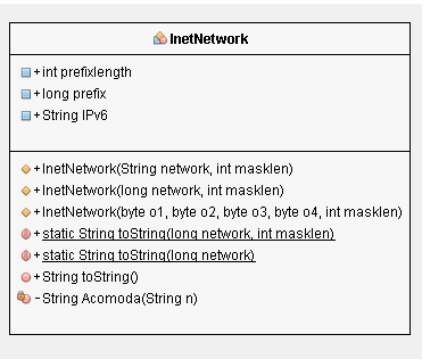

Figura 5.23: UML de InetNetwork

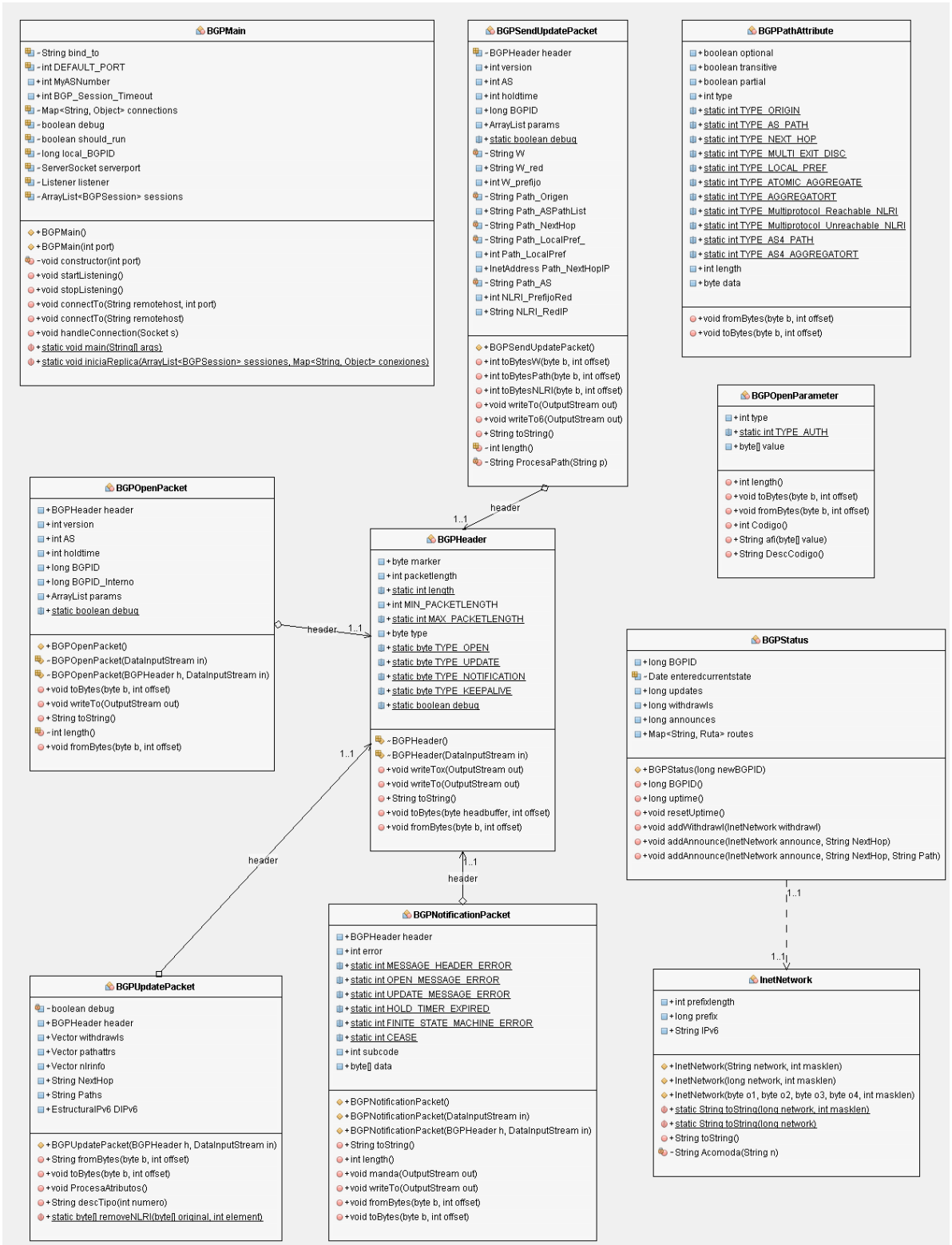

Figura 5.24: UML de las clases principales de la aplicación

5.4. Resultados de Optimizar Rutas

Experimentalmente se han detectado problemas en los principales proveedores mexicanos de servicios de Internet. Para tener redundancia en los servicios es adecuado tener dos o más ISPs como ya hemos explicado en este trabajo. Estos proveedores cometen algunos errores como por ejemplo omitir algunas rutas hacia otros ISPs de México, lo que provoca que el tráfico, que podría utilizar redes nacionales, pase a Estados Unidos y nuevamente regrese a México. En la práctica se probó que el promedio de la latencia se incrementa en 40 ms.

En la Figura 6.1 se ven los diferentes sistemas autónomos por cada país que se pueden optimizar en la red de MetroRed y en la Figura 5.25 se muestran los tiempos antes y después de optimizar una ruta que involucraba a MetroRed y AS6503. Ambos proveedores enviaban el tráfico a “Cogent Communications” en USA, pese a que tenían un enlace directo entre ellos. En un análisis de todas las rutas, se puede encontrar que es posible optimizar el 12 % de todos los prefijos de IPv4 y de hasta un 10 % en todos los prefijos en IPv6 como se pueden apreciar en las Tablas 5.16 y 5.17. Aunque parece muy bajo el porcentaje de rutas a optimizar, si tomamos en cuenta cúantos paquetes de datos son enviados por estas rutas no optimizadas, el efecto positivo en el desempeño de las comunicaciones sobre esas rutas crece sustancialmente.

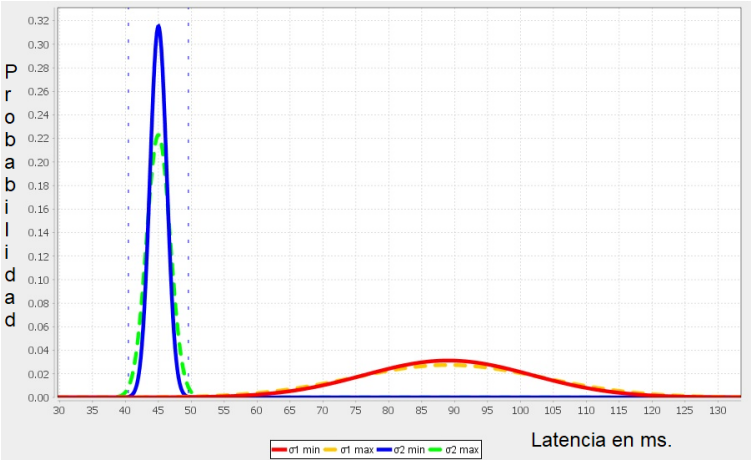

Figura 5.25: Mejora en Latencias antes y después de optimizar rutas.

Se redujo de 90ms a 45ms el promedio de latencia al recortar los saltos al destino.

Un proceso similar al anteriormente descrito se realizó sobre la red de la Universidad Panamericana, en la que se almacenó durante quince días las estadísticas de tráfico que se originaba en el campus, y se alcanzó un total de 787,956 direcciones IP.

De esta base de datos se tomó una muestra aleatoria de 1010 destinos IP (30 % de los segmentos de red diferentes en un día para garantizar el 99 % representatividad). Un total de 383 no respondieron a un enlace con el comando "Ping" (no se tenía acceso a esas direcciones IP) por lo que fueron descartadas, además 4 de esas direcciones no fueron encontradas en la tabla de rutas BGP por tratarse de direcciones internas de la Universidad Panamericana. Con los 623 restantes se procedió a experimentar para determinar sus latencias.

Estos experimentos se realizaron utilizando los tres proveedores de servicios con los que contamos en nuestro AS y sin tomar en consideración la mejor ruta que propone el protocolo BGP. Para nuestros tres proveedores de ISP la distribución de mejores rutas puede observarse en la Figura 5.26.

De cada una de la 623 IPs en el experimento y por cada uno de los tres proveedores, almacenamos el tiempo mínimo, el tiempo máximo y el promedio de envío de cinco paquetes de ICMP. El tiempo menor obtenido para este experimento fue de 1 ms con los proveedores A y B. El tiempo mayor obtenido fue de 1648 ms con el proveedor C. Los experimentos mostraron que el promedio de los cinco paquetes de ICMP, las mejores rutas (menor latencia) por cada proveedor quedan:

Se puede ver en las Figuras 5.26 y 5.27, que según BGP el proveedor B tiene sólo el 9 % de las mejores rutas, pero si se consideran latencias, su porcentaje de mejores rutas crece al 32 % (de 57 rutas a 199). En este análisis en que los tres proveedores cuentan con igual ancho de banda utilizar la ruta con base al parámetro de la menor latencia, obtiene un resultado con mejor distribución de las rutas, aunque es oportuno aclarar que no necesariamente es un balance del tráfico, pues depende del ancho de banda del tráfico de cada destino en particular.

No todas las direcciones IP respondieron al ICMP, por lo que sólo a 623 rutas se les pueden comparar la latencia. Estas rutas originalmente se distribuyen de manera no balanceada, ya que el 50 % de esos destinos prefieren a un solo proveedor. Se puede aprovechar el cambio de rutas por proveedores de menor latencia para tener un tráfico más equitativo considerando los anchos de banda. De las 623 rutas que respondieron al "ping", cuando se dirigió el tráfico por la ruta hacia el destino con menor latencia, el resultado global con tres proveedores fue mejor, al quedar los proveedores con 40 %, 32 % y 21 % de mejores rutas respectivamente. Sólo en un 7 % de rutas (44 rutas) la latencia es diferente en por lo menos dos proveedores ISP, por lo que se pueden dejar las rutas tal y como el protocolo BGP las configura o modificarlas para balancear mejor las salidas.

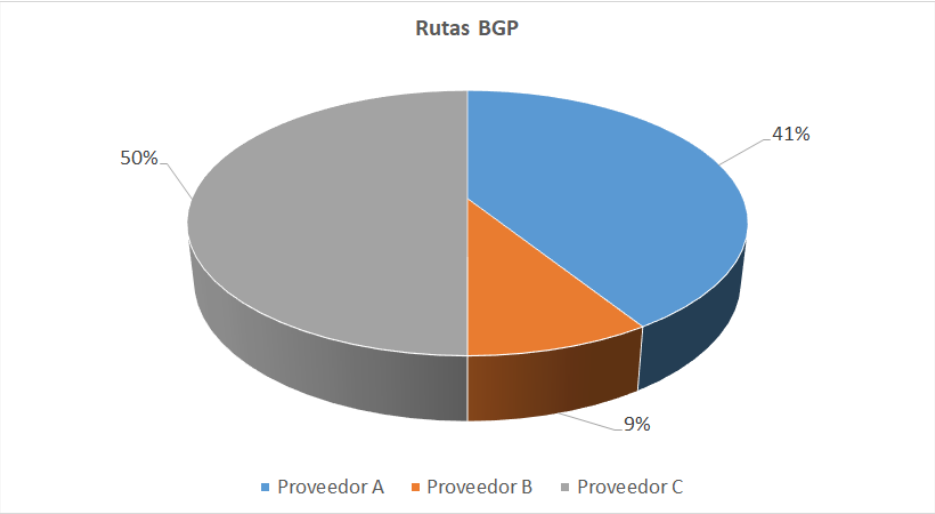

Figura 5.26: Distribución de las rutas de 623 destinos IPv4 según BGP

Porcentajes de cada uno de los tres proveedores.

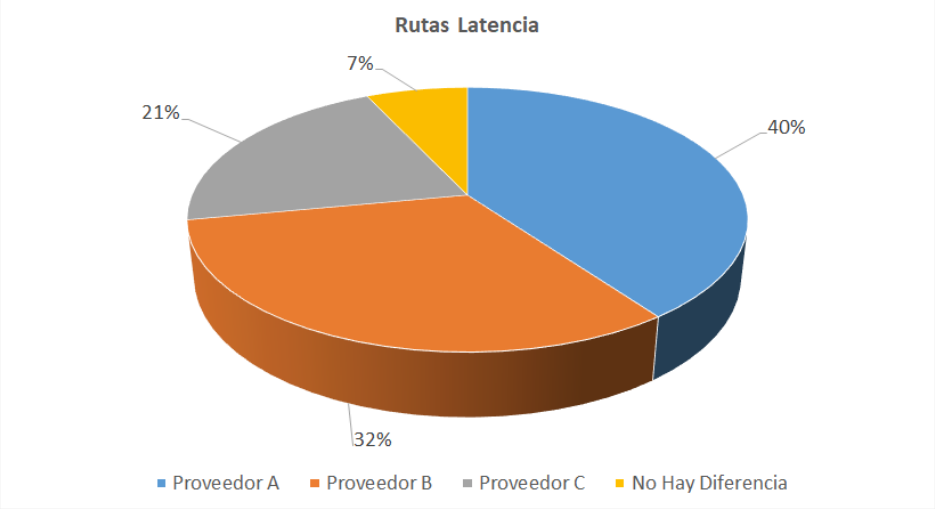

Figura 5.27: Distribución de las rutas de 623 destinos IPv4 por latencia.

Porcentajes de en qué situación se da la menor latencia.

Al utilizar el algoritmo propuesto en esta tesis, en un análisis de las latencias por los diferentes enlaces de los tres proveedores, se determinó que si no se tuviera la optimización sobre el protocolo BGP, el tiempo promedio de las latencias hacia todos los destinos sería de 223 ms, alcanzando un latencia máxima promedio de 264 ms. Utilizando las rutas no optimizadas sobre BGP el promedio de la latencia se reduce a 202 ms (un 10 por ciento menos) y la latencia máxima promedio disminuye 228.36 ms. (un 14 por ciento menos). Al incorporar nuestro algoritmo de utilizar la mejor ruta la latencia promedio obtenida fue de 167 ms y promedio de la latencia máxima de 179 ms lo que implica una disminución todavía mayor en ambos indicadores de un

26 por ciento y 33 por ciento respectivamente.

5.4.1. Resumen de Resultados de los experimentos de optimización

En la Tabla 5.16 se pueden ver las rutas a optimizar de los diferentes ISP que se evaluaron. El promedio de rutas a optimizar en IPv4 es de 12 % y en IPv6 de 10 %. Resultados similares se encontraron en los primeros experimentos usando los 3 proveedores de Internet que tiene la Universidad Panamericana, y debido a que desde que se identificaron estas rutas, se han reportado a los proveedores, se ha logrado que estas rutas fueran corregidas. Estos últimos resultados se presentan en la Tabla 5.17 donde también se muestran la cantidad de vecinos y el promedio del tamaño del camino.

Proveedor	ISP1		ISP2		ISP3		ISP4		ISP5	
Protocolo	IPv4	IPv6	IPv4	IPv6	IPv4	IPv6	IPv4	IPv6	IPv4	IPv6
Rutas	682943	57383	752598	84401	731188	58974	783411	82480	728271	92236
No óptimas	81953	7429	91741	7858	92787	5567	105212	8330	89993	6928
% a Optimizar	11.99	12.94	12.19	9.31	12.69	9.44	13.43	10.10	12.36	7.51
Prom. Path	4.504	4.176	4.539	4.456	4.386	4.030	4.631	3.921	4.441	4.133

Tabla 5.16: Resumen de las rutas con posibilidad de optimizar por ISP.

Para dimensionar el beneficio de este nuevo método, se puede comparar contra otras soluciones como la de compresión de datos o de túneles directos que no impactan a más del 1 % del tráfico comparado con el método propuesto en esta tesis que puede llegar a más del 10 % del tráfico de red, [Hamdoun et al., 2021] aunque la compresión busca otros beneficios como el de aprovechar mejor el ancho de banda.

Un beneficio adicional a la reducción de la latencia como resultado de disminuir el salto en las rutas, es la mejora de la varianza, lo que indica que los caminos son más estables y al pasar por menos redes diferentes, se incrementa la disponibilidad del servicio a ser menos propenso a fallas.

Con los siguientes ejemplos, se muestra la importancia de optimizar cualquier ruta, ya que esto permite mejorar la latencia para esas rutas en concreto y permite, al reducir el tiempo que esos paquetes están en la red, mejorar además los anchos de banda y beneficiar también otras rutas. En los experimentos realizados en esta tesis se pueden llegar a optimizar en promedio el 12 % de las rutas, que, aunque pueda parecer

Proveedor	AS6503		AS18734		AS22566		UP	
Protocolo	IPv4	IPv6	IPv4	IPv6	IPv4	IPv6	IPv4	IPv6
Rutas	695253	49092	681682	47716	683538	48262	698417	49636
No óptimas	3812	1159	44363	563	148	2	98	7
Vecinos	40	9	181	31	13	4	3	3
% a Optimizar	0.55	2.36	6.50	1.18	0.02	0.004	0.014	0.014
Prom. Path	4.195	4.503	4.539	3.964	4.724	4.280	**4.108**	**3.832**

Tabla 5.17: Resumen de las rutas optimizadas por ISP.

pequeño este porcentaje, fija las bases para futuras investigaciones y es una primera mejora. Las soluciones de optimización de redes ayudan a los sistemas sobre todo en ambiente híbridos, donde los respaldos, copia de seguridad y muchos servicios en la nube y distribuidos requieren el uso de Internet. Al mover datos hacia o desde servicios de respaldo remotos o basados en la nube, la optimización de la latencia, es cada vez más importante.

A continuación mostraremos algunos ejemplos para visualizar lo relevante este 10 % de mejora:

1. Según reportes del [Hernández, 2020, INEE, 2019] en México, 25 millones de estudiantes de la SEP, toman clases a distancia y hay más de un millón doscientos mil docentes. El mercado de las telecomunicaciones está dominado por América Móvil (Telmex y Telcel) con un 60 % del mercado, y Grupo Televisa con un 12.3 % [Opportimes, 2020]. Si se considera a los usuarios de este último, se tienen 3.22 millones de usuarios, que podrían no tener rutas óptimas y, por lo tanto, una latencia mayor en sus videoconferencias, se puede dimensionar la cantidad de usuarios beneficiados con estas mejoras.

   De los 4 principales beneficios de las videoconferencias, reducción de costos, trabajo colaborativo, mejorar la comunicación y hacer más eficiente el trabajo en equipo, los 2 últimos, requieren latencias pequeñas para evitar pedida de atención durante la videoconferencia y para aprovechar mejor el tiempo de conexión.

   La Figura 5.28 y las Figuras 5.29 y 5.30 muestran las estadísticas que tienen las herramientas de Google Meet y de Zoom para monitorear las latencias pues es el parámetro directamente relacionado

con la calidad de la videoconferencia.

2. Según un estudio de Deloitte [Deloitte, 2020] solicitado por Google, donde demuestra que una mejora del 0.1 segundos, genera que los consumidores compren un 10 % adicional, y que latencias de más de 100 ms ya son percibidas por los clientes. Estas mejoras, repercuten, en el caso de comercio en línea, además del aumento en las compras, mejora la experiencia de los usuarios.

   Google ofrece una herramienta para evaluar la velocidad de las páginas, PageSpeed Insights en donde califica de cero a cien, el rendimiento de una página web, considerando principalmente: El tiempo que tarde en mostrarse la página, el rendimiento de la carga de la página, el tamaño de las imágenes y la estabilidad visual. El rendimiento de la página, especifica que debe de cargarse completamente en menos de 2.5 segundos, por lo que la latencia afecta la cantidad de objetos que contenga esa página y por supuesto su calificación. El ejemplo que se muestran en las Figura 5.31 permite comparar dos sitios diferentes para comparar sus parámetros.

   Esta herramienta, revisa el rendimiento y muestra la distribución de varios parámetros de los últimos 28 días. Estos parámetros miden tres aspectos de la experiencia del usuario: carga, interactividad y estabilidad visual. El valor de “First Contentful Paint” representa, el tiempo que toma en el que se muestra el primer texto o la primera imagen, por lo que valores buenos se consideran menores a 1 segundo, que se pueden mejorar entre 1 y 2 segundos y desempeño pobre si el tiempo es mayor a 2 segundos, por lo que el 1er ejemplo la calificación es de color rojo y el segundo es verde. El tiempo de “Largest Contentful Paint” es el relacionado con el del mayor elemento por lo que este valor indica el tiempo que se tarda en dibujar el texto o la imagen de mayor tamaño. El “First Input Delay” es el parámetro que está más relacionado con la latencia y consiste en el tiempo que toma recibir los primeros datos. En este ejemplo ambas referencias están por debajo de los 100 ms recomendados para proporcionar una experiencia al usuario fluida. Sitios muy lejanos del usuario, pueden afectar más este parámetro, que la respuesta del servidor ya que la demora debido al enlace, fácilmente pueden agregar más de 50 ms. a este tiempo. “Cumulative Layout Shift” representa la estabilidad visual que recibe el usuario. Este valor debe de estar debajo de 0.1.

3. En un escenario de control remoto utilizando Internet, se envían más de 600 instrucciones por cada

sesión de control. Si estas instrucciones viajan por una ruta no optimizada, su latencia promedio puede ser de hasta 150 ms mayor que por una ruta optimizada, lo que equivale a una pérdida de tiempo de más de un minuto y medio. Adicionalmente, una latencia menor, mejora la retroalimentación y se generan menos errores en ambientes de telecomunicaciones más fluidos. Una excavadora hidráulica promedio tiene una velocidad de 5.7 km/h [CAT2020, 2020] o de 20.52 m/seg., por lo que una latencia de 100 ms con respecto a lo que se recibe y lo que se transmite equivale a una diferencia de desplazamiento de 2 metros, por lo que es importante investigaciones como [Uitto et al., 2019, Isto et al., 2020] en este tema.

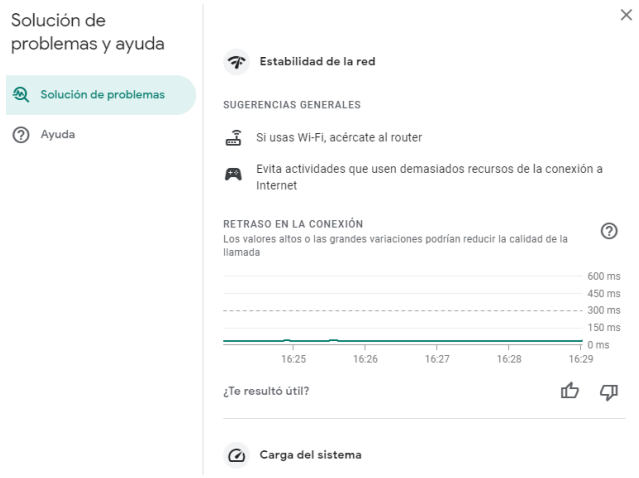

Figura 5.28: Estadísticas de Google Meet

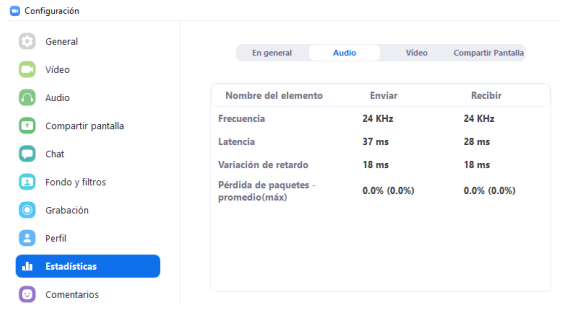

Figura 5.29: Estadísticas de Audio en Zoom

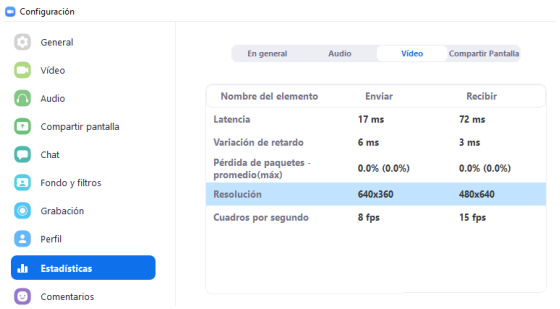

Figura 5.30: Estadísticas de Video en Zoom

La aplicación desarrollada en esta Tesis, tiene la capacidad de corregir las rutas que no son óptimas, agregando las rutas descubiertas con los parámetros adecuados para que tenga prioridad esas rutas. Adicionalmente, los ISP pueden solucionar estas rutas de fondo, atacando las causas del por qué no esas rutas no

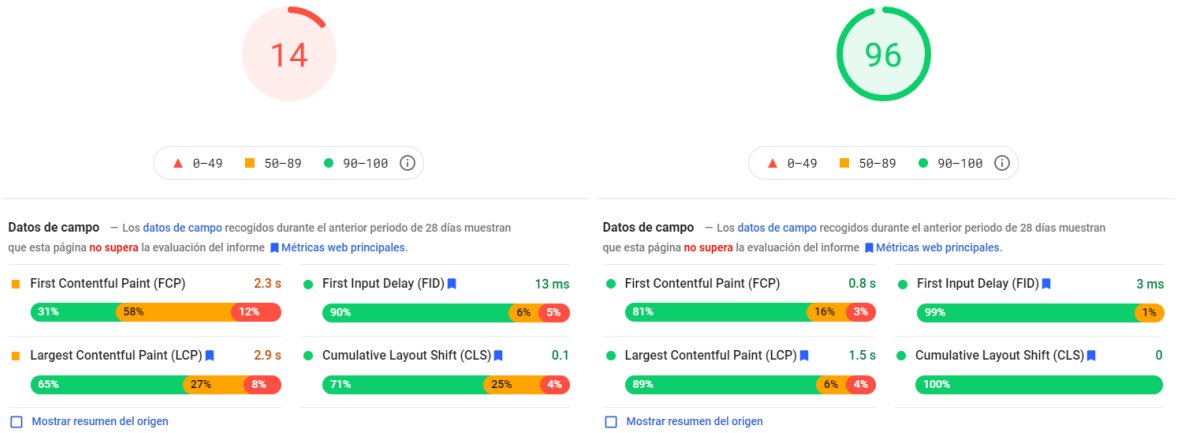

Figura 5.31: Ejemplos de PageSpeed con calificaciones de 14 y 96

eran las adecuadas, que en la mayoría de los casos se deben a omisiones en la configuración de los equipos que originan esas rutas. El beneficio de la optimización puede abarcar otras rutas, procesamiento en los ruteadores y una menor utilización en la red lo que puede incluso reducir costos de los enlaces. En el caso de la experiencia con la presente Tesis se nota en las rutas nacionales que tenían a ir a EUA y regresar, que se pueden beneficiar si solo el tráfico se propaga de manera local, es decir, solo nacional.

5.4.2. Comparativo de la solución propuesta con OpenDaylight

OpenDayLight, es una solución para SDN [Badotra and Panda, 2019] muy robusta y completa. El módulo de BGP es solo un componente o módulo [Cajas and Budanov, 2020] de todos los que se pueden instalar. Es de código abierto, es pate de "Linux Foundation"[Biddle, 2019], está desarrollada en Java, se basa en APIs y configuración mediante archivos XML. Esta soportada por la mayoría de fabricantes de equipos de redes y cumple con todos los protocolos estándares. En apenas algunos años de desarrollo cuenta con más de 200 socios involucrados en el proyecto [Project, 2020a], dentro de las cuales se encuentras Universidades y centros de investigación como Stanford y Berkeley. Sus principales semejanzas es que están escritas en Java, buscan implementar los últimos estándares y son modulares.

5.4.3. Comparativo de la solución propuesta con λBGP

Esta solución, presentada en el 2020, [Hart et al., 2020], es una herramienta escrita en Haskell, lenguaje de programación puramente funcional [Bird and Gibbons, 2020], con base en SDN y un conjunto de API de

control de red abiertas y genéricas con la finalidad de mejorar la funcionalidad de la red y proporcionar mayor flexibilidad en la gestión de la red.

Sus similitudes con las propuestas en esta tesis, son que ambos utilizan BGP, SDN, y permiten la modificación de las rutas. Aunque se utilizan aproximaciones diferentes, ambos permiten la integración de mecanismos similares en el corazón del algoritmo de selección de ruta. También permiten, el procesamiento de nuevas rutas en paralelo. Las principales diferencias, adicional a los lenguajes de programación son: λBGP no soporta de momento IPv6, se enfoca más en la gestión de la red, y orientado al uso de API.

5.4.4. Comparativo de la solución propuesta con xBGP

Esta propuesta, similar a λBGP es una capa de software, o librería, con funciones específicas en forma de API. Los autores pretenden acelerar la innovación facilitando la implementación de políticas diferentes en BGP [Wirtgen et al., 2020] y otros protocolos.

A diferencia de esta tesis, tiene un enfoque más a políticas, seguridad, filtrado, rendimiento, etc., que en la optimización de las rutas y se proporciona en un esquema de máquina virtual.

Capítulo 6

Conclusiones

En esta tesis doctoral, se resume la investigación realizada para optimizar el acceso a Internet utilizando SDN. Se basa en que las rutas de BGP, aunque son óptimas desde el punto de vista del "número de saltos", no lo son desde el punto de vista de la latencia, por lo que se requiere:

1. Una forma de comparar rutas.
2. Un algoritmo para encontrar rutas no óptimas
3. Un mecanismo para modificar las rutas existentes por las rutas optimizadas.

Los resultados obtenidos demuestran que la forma de comparar rutas calculando una variación de las latencias mínimas y máximas, permiten tener un parámetro numérico de la diferencia en cada una de las rutas de conexión en lo que se refiere a la latencia del enlace. El algoritmo para encontrar rutas no óptimas se probó con varios ISP, encontrándose que aproximadamente 100,000 rutas de 830,000 en IPv4 equivalente al 12 % de las rutas se pueden mejorar optimizando los caminos.

Todas las aplicaciones utilizadas y todos los desarrollos, se implementaron tanto para el protocolo IPv4 e como para el protocolo IPv6. Mantener "Dual Stack", o sea ambos protocolos, permiten tener compatibilidad hacia el futuro y permite demostrar que IPv6 ya es un protocolo maduro y que se utiliza en el 60 % del tráfico en Internet.

6.1. Comprobación de Hipótesis

En la sección 1.3.1, se plantea la primera hipótesis: **H1 El algoritmo "Analiza-Rutas" encuentra las mejores rutas en una red con base en la información de la topología completa y de los vecinos directos.** y en la sección 4.2 se desarrolla el algoritmo para comparar las latencias con cinco mediciones usando intervalos de confianza. Esto permite tener una estimación de si efectivamente una ruta es de menor latencia que otra o no presenta beneficio alguno sobre las demás rutas. Por lo que se demuestra

que sí se puede encontrar la forma de analizar si una ruta es mejor que las demás utilizando el método propuesto en este trabajo doctoral.

En el caso de esta tesis, donde el enfoque es de optimizar latencias, no se consideran otras soluciones, por ejemplo, el de deduplicación y compresión de la información [Gero, 2020], cuyo propósito es el de disminuir el ancho de banda necesario y no ayuda a la latencia directamente (Y se puede implementar sin necesidad de SDN), ni de soluciones que mejoren la disponibilidad de los enlaces o estén relacionados con la seguridad [Al-Darrab et al., 2020, Al-Hayajneh et al., 2020], y aunque las herramientas utilizadas en esta tesis se podrían utilizar para esos fines, el alcance es tan amplio y en este caso, solo se enfoca en la reducción de la latencia.

En la sección 1.3.1, se plantea la segunda hipótesis: **H2 El método para mejorar las tablas de rutas en una red es práctico de implementar y mejora la latencia de las rutas optimizadas.** La aplicación desarrollada y probada en diferentes tamaños de redes, demuestra que una vez que se determinan las rutas a optimizar, es posible corregir esas rutas. Los beneficios en este caso son los menores tiempos de trasmisión de información a través de las rutas optimizadas. En simulaciones realizadas también se pudo comprobar que las soluciones propuestas pueden funcionar a futuro incluso aunque el Internet duplique su tamaño actual. Esto está comprobado en la sección Resultados de Optimizar Rutas 5.4 (pág. 87) y un ejemplo concreto se encuentra en el anexo B.2.1 (pág. 197)

La tercera hipótesis: **H3 La optimización de las rutas, entre el origen y el destino, en una red depende del número de saltos entre sistemas autónomos; lo que reduce la distancia y, a su vez, impacta en la disminución de la latencia.** Esta hipótesis plantea la relación entre el número de saltos y la distancia. Como la latencia es una función directa de la distancia, al acortar el número de saltos, se está reduciendo la distancia que recorre la información en la red, y por lo mismo también se reduce el tiempo que le toma transitar desde el origen hasta destino. Esto se demostró en el capítulo 4 y 5, donde la latencia total descrita por la fórmula 4.1 (pág. 47), y la relación entre distancia y latencia se mostró con el experimento factorial 5.1 y el número de saltos es el principal factor en beneficiarse con el algoritmo que plantea esta tesis.

6.2. Resultados obtenidos

Se detectó en los experimentos realizados que los principales proveedores mexicanos o ISPs, omiten el anuncio de rutas hacia otros proveedores mexicanos, lo que genera que tráfico que pudiera quedarse de manera local, salga a proveedores en Estados Unidos y regrese a México con las desventajas en latencias que conlleva salir y entrar a los proveedores nacionales. La aplicación soluciona todas estas desventajas y permite obtener un reporte para entender cuál es la causa raíz de estas rutas no optimizadas. A manera de ejemplo, una ruta a Estados Unidos, suele tener una latencia entre 37 y 50 milisegundos, si esa ruta no debió salir de México, donde las latencias son entre 4 y 15 ms, se tiene un beneficio de 60 % al 90 % del tiempo total.

Adicionalmente a esos reportes, se han revisado rutas internacionales que permiten determinar aquellas que pasan por países innecesariamente. Corregir estas rutas, además del beneficio en la disminución de la latencia, puede reducir riesgos de seguridad.

El desarrollo de una aplicación, que cumpliendo con los estándares de BGP, es capaz de interactuar con los ruteadores, permite, obtener la información real de ruteo y procesarla para optimizar con el beneficio de mejorar las latencias. Esta aplicación, de mucha utilidad para las empresas dedicadas a proveer servicios de Internet, permite iniciar otras líneas de investigación, por ejemplo, utilizando IA.

Como beneficio adicional a la reducción de la latencia, también se aprecia una mejora en la varianza.

6.3. Divulgación de la Investigación y Aplicación en la Industria

A partir del 2016, se han publicado cuatro artículos, de los cuales tres aparecen en revistas indexadas en Scopus: dos en Science Direct y uno en Springer en una revista Q2.

- Elguea, L. M., Martinez-Rios, F. O., and Ramos, J. C. (2016). Latency analysis in IPv4/IPv6 Data Network. International Journal of P2P Network Trends and Technology (IJPTT), pages V25:6–10 Apr–2016.

  En este artículo se presenta el análisis factorial (sección 5.1) y se hace el comparativo de IPv4 e IPv6.

- Elguea, L. M. and Martinez-Rios, F. (2017). An efficient method to compare latencies in order to obtain the best route for SDN. Procedia Computer Science, 116:393 – 400. Discovery and innovation

of computer science technology in artificial intelligence era: The 2nd International Conference on Computer Science and Computational Intelligence (ICCSCI 2017).

En esta publicación se muestra el método copara comparar las latencias (capitulo 4).

- Elguea, L. M. and Martinez-Rios, F. (2018). A new method to optimize bgp routes using sdn and reducing latency. Procedia Computer Science, 135:163 – 169. The 3rd International Conference on Computer Science and Computational Intelligence (ICCSCI 2018) : Empowering Smart Technology in Digital Era for a Better Life.

  Aqui se describe el algoritmo completo, sección 4.3, los beneficios de optimizar las rutas, sección 5.1.6, el proceso para optimizar rutas (comparativo en 4.3.1) y como determinar las rutas a optimizar.

- Elguea, L. M. and Martinez-Rios, F. (2019). New metrics to modify bgp routes based on sdn. Wireless Networks, doi 10.1007/s11276-019-02025-3.

  En este artículo más reciente, se proponen nuevas métricas que se pueden encontrar con herramientas como SDN en redes WAN para determinar parámetros, como en que países están registrados los sistemas autónomos.

Adicionalmente a estas publicaciones, durante el desarrollo de la Tesis se elaboró una aplicación para la industria que se ha probado con éxito en varias redes nacionales de gran tamaño. La aplicación implementa los algoritmos utilizados en esta tesis y permitieron a una empresa que pertenece al ramo de las telecomunicaciones y es un proveedor con alcance nacional lo siguiente:

1. Diagnóstico preliminar y recopilación de información de rutas de enlace en internet.
2. Detección de errores en las rutas utilizando SDN y el algoritmo desarrollado, para calcular métricas de desempeño basado en la distancia más corta y la menor latencia.
3. Elaboración de nuevas rutas de interconexión a internet para un mejor desempeño de los enlaces.

6.3.1. Trabajos futuros

Claramente el despliegue de SDN en los primeros años de la segunda década de este siglo, crecerá enormemente. Redes como la 5G están basadas en esta arquitectura [Lien et al., 2019, Trivisonno et al., 2015, Sun et al., 2015, Jayakody et al., 2019], y las redes existentes se están migrando a estas soluciones.

Existe una propuesta de estándar con base en el RFC7854 que se encuentra ya implementado en algunas aplicaciones de SDN como OpenDaylight (ODL BMP plugin). Desarrollar una aplicación pequeña que lo implemente, puede ser de mucha utilidad para cualquier ISP ya que la propuesta de esta tesis, permite, además del monitoreo de las rutas, reportes estadísticos, que claramente permitirían un análisis que puede ayudar a optimizar las rutas.

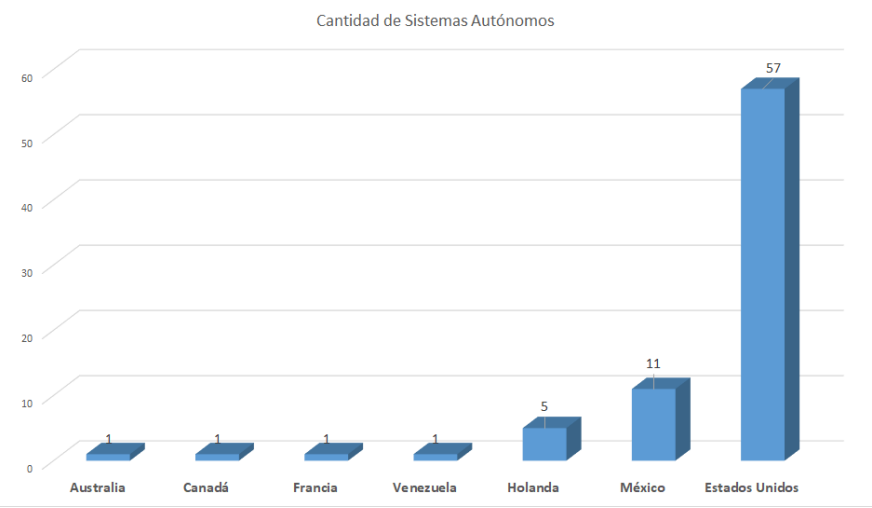

Figura 6.1: Posibles rutas a optimizar por número de Sistemas Autónomos.

La Tabla B.1 muestra sistemas autónomos mexicanos conectados a un ISP nacional, también se muestras el número de segmentos de cada AS que no están optimizados. En este caso, los tiempos que se obtienen al optimizar estas rutas, son un 50 % menor a los tiempos que se tienen en condiciones normales. Esto se debe a que cada segmento que no está correctamente anunciado, ocasiona que los paquetes tengan que llegar a Estados Unidos y regresar para alcanzar su objetivo. En la Figura 6.1 se pueden ver, en proporción, los 11 sistemas autónomos en México que definitivamente conviene optimizar.

Bibliografía

[Abbas et al., 2020] Abbas, K., Afaq, M., Ahmed Khan, T., Rafiq, A., Iqbal, J., Ul Islam, I., and Song, W.-C. (2020). An efficient sdn-based lte-wifi spectrum aggregation system for heterogeneous 5g networks. **Transactions on Emerging Telecommunications Technologies**, page e3943.

[Al-Darrab et al., 2020] Al-Darrab, A., Al-Darrab, I., and Rushdi, A. (2020). Software-defined networking load distribution technique for an internet service provider. **Journal of Network and Computer Applications**, 155:102547.

[Al-Hayajneh et al., 2020] Al-Hayajneh, A., Bhuiyan, Z. A., and McAndrew, I. (2020). Improving internet of things (iot) security with software-defined networking (sdn). **Computers**, 9(1):8.

[Alba et al., 2004] Alba, E., Luque, G., and Troya, J. M. (2004). Parallel lan/wan heuristics for optimization. **Parallel Computing**, 30(5):611–628.

[Amini et al., 2002] Amini, L. D., Shaikh, A., and Schulzrinne, H. G. (2002). Issues with inferring internet topological attributes. In **Internet Performance and Control of Network Systems III**, volume 4865, pages 80–91. International Society for Optics and Photonics.

[Amiri et al., 2020] Amiri, M., Osman, H. A., and Shirmohammadi, S. (2020). Resource optimization through hierarchical sdn-enabled inter data center network for cloud gaming. In **Proceedings of the 11th ACM Multimedia Systems Conference**, pages 166–177.

[Anderson et al., 1993] Anderson, T. E., Owicki, S. S., Saxe, J. B., and Thacker, C. P. (1993). High-speed switch scheduling for local-area networks. **ACM Transactions on Computer Systems (TOCS)**, 11(4):319–352.

[Andreev and Kanto, 2004] Andreev, A. and Kanto, A. (2004). Conditional value-at-risk estimation using non-integer values of degrees of freedom in student's t-distribution. **The Journal of Risk**, 7(2):1.

[Auroux et al., 2015] Auroux, S., Draxler, M., Morelli, A., and Mancuso, V. (2015). Dynamic network reconfiguration in wireless densenets with the crowd sdn architecture. **The University for the Information Society**.

[Badotra and Panda, 2019] Badotra, S. and Panda, S. N. (2019). Evaluation and comparison of opendaylight and open networking operating system in software-defined networking. **Cluster Computing**, pages 1–11.

[Baldonado et al., 2008] Baldonado, O., Finn, S., Karam, M., Lloyd, M., Madan, H., McGuire, J., and Villaverde, J. (2008). Method and apparatus for performance and cost optimization in an internetwork. US Patent 7,406,539.

[Banga et al., 1997] Banga, G., Douglis, F., Rabinovich, M., et al. (1997). Optimistic deltas for www latency reduction. In **Proc. 1997 USENIX Technical Conference, Anaheim, CA**, pages 289–303.

[Barracuda-Networks, 2013] Barracuda-Networks (2013). Using an ng firewall for wan optimization. `https://www.barracuda.com/assets/docs/White_Papers/Barracuda_NG_Firewall_WP_Wan_Optimization.pdf` Consultado el 18/marzo/2017.

[Beloglazov et al., 2012] Beloglazov, A., Abawajy, J., and Buyya, R. (2012). Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. **Future generation computer systems**, 28(5):755–768.

[Beloglazov and Buyya, 2010] Beloglazov, A. and Buyya, R. (2010). Energy efficient resource management in virtualized cloud data centers. In **Proceedings of the 2010 10th IEEE/ACM international conference on cluster, cloud and grid computing**, pages 826–831. IEEE Computer Society.

[Biddle, 2019] Biddle, B. (2019). Linux foundation is eating the world. **Available at SSRN 3377799**.

[Bird and Gibbons, 2020] Bird, R. and Gibbons, J. (2020). **Algorithm Design with Haskell**. Cambridge University Press.

[Black, 1991] Black, U. (1991). **OSI: A Model for Computer Communications Standards**. Prentice-Hall.

[Breyfogle III, 1992] Breyfogle III, F. W. (1992). **Statistical methods for testing, development, and manufacturing**. John Wiley & Sons.

[Brief, 2014] Brief, O. S. (2014). Openflow-enabled sdn and network functions virtualization. **Open Netw. Found**, 17:1–12.

[Briscoe, 2000] Briscoe, N. (2000). Understanding the osi 7-layer model. **PC Network Advisor**, 120(2).

[Bruegge and Dutoit, 2009] Bruegge, B. and Dutoit, A. H. (2009). Object–oriented software engineering. using uml, patterns, and java. **Learning**, 5(6):7.

[Bruno Nunes et al., 2014] Bruno Nunes, A., Marc Mendon, C., Xuan Nam, N., Katia, O., and Thierry, T. (2014). A survey of software - defined networking: Past, present, and future of programmable networks. **Communications Surveys and Tutorials, IEEE Communications Society, IEEE**, pages 1617 – 1634. <10.1109/SURV.2014.012214.00180>. <hal-00825087v5>
`http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6739370` Consultado el 10/octubre/2020.

[Buyya et al., 2010] Buyya, R., Beloglazov, A., and Abawajy, J. (2010). Energy-efficient management of data center resources for cloud computing: a vision, architectural elements, and open challenges. **arXiv preprint arXiv:1006.0308**.

[Cabaj et al., 2019] Cabaj, K., Gregorczyk, M., Mazurczyk, W., Nowakowski, P., and Żórawski, P. (2019). Network threats mitigation using software-defined networking for the 5g internet of radio light system. **Security and Communication Networks**, 2019.

[Cajas and Budanov, 2020] Cajas, C. D. and Budanov, D. O. (2020). Sdn applications and plugins in the opendaylight controller. In **2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)**, pages 9–13. IEEE.

[CAT2020, 2020] CAT2020 (2020). Escabadora cat 320.
`https://s7d2.scene7.com/is/content/Caterpillar/CM20180604-55406-39118` Consultado el 5/marzo/2021.

[Chaitanya and Kishore, 2010] Chaitanya, K. K. and Kishore, C. R. (2010). An approach to shortest path technique for bgp using ospf. **Int J Comput Sci Inform Technol**, 1(5):389–391.

[Chase et al., 2014] Chase, J., Kaewpuang, R., Yonggang, W., and Niyato, D. (2014). Joint virtual machine and bandwidth allocation in software defined network (sdn) and cloud computing environments. In **2014 IEEE International Conference on Communications (ICC)**, pages 2969–2974. IEEE.

[Chavez, 2010] Chavez, C. (2010). Sistema de monitoreo abierto. `http://mrtg.upmx.mx/` Consultado el 19/octubre/2020.

[Chen et al., 2015] Chen, T., Matinmikko, M., Chen, X., Zhou, X., and Ahokangas, P. (2015). Software defined mobile networks: concept, survey, and research directions. **IEEE Communications Magazine**, 53(11):126–133.

[Cicic and Bryhni, 2010] Cicic, T. and Bryhni, H. (2010). device and system for selective forwarding. US Patent App. 12/828,835.

[Cisco, 2008] Cisco (2008). Casos prácticos de BGP. `https://www.cisco.com/c/es_mx/support/docs/ip/border-gateway-protocol-bgp/26634-bgp-toc.html` Consultado el 10/octubre/2020.

[Cisco, 2010a] Cisco (2010a). Comparing ipv4 and ipv6 throughput and latency on cisco router platforms. **Executive Summary**, 1:1–2. `http://www.cisco.com/web/strategy/docs/gov/IPv6perf_es.pdf` Consultado el 10/octubre/2020.

[Cisco, 2010b] Cisco (2010b). Performance-comparison testing of ipv4 and ipv6 throughput and latency on key cisco router platforms. **A Summary of Findings**, 1:1–25. `http://www.cisco.com/c/dam/en/us/products/collateral/ios-nx-os-software/enterprise-ipv6-solution/IPv6perf_wp1f.pdf` Consultado el 10/octubre/2020.

[Cisco, 2011] Cisco (2011). Bgp commands: M through neighbor soo - cisco. download from vendor site. `http://www.cisco.com/c/en/us/td/docs/ios/iproute_bgp/command/reference/irg_book/irg_bgp3.pdf` Consultado el 15/abril/2017.

[Cisco, 2015] Cisco (2015). Cisco wireless optimization service. `https://www.cisco.com/c/dam/en_us/services/downloads/wireless-optimization-service-aag.pdf` Consultado el 10/octubre/2020.

[Cisco Networking Academy, 2014] Cisco Networking Academy (2014). **Routing Protocols Companion Guide**, volume 1. Cisco Press, 1 edition.

[Citrix, 2014] Citrix (2014). An introduction to software defined networking.

[Cvijetic et al., 2013] Cvijetic, N., Angelou, M., Patel, A., Ji, P. N., and Wang, T. (2013). Defining optical software-defined networks (sdn): From a compilation of demos to network model synthesis. **Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference (OFC/NFOEC), 2013**, pages 1–3.

[de Winter, 2013] de Winter, J. C. (2013). Using the student's t-test with extremely small sample sizes. **Practical Assessment, Research & Evaluation**, 18(10):1–12.

[Delling et al., 2009] Delling, D., Sanders, P., Schultes, D., and Wagner, D. (2009). Engineering route planning algorithms. In **Algorithmics of large and complex networks**, pages 117–139. Springer.

[Deloitte, 2020] Deloitte (2020). Milliseconds make millions. `https://www2.deloitte.com/content/dam/Deloitte/ie/Documents/Consulting/Milliseconds_Make_Millions_report.pdf` Consultado el 2/marzo/2021.

[Dewanto et al., 2018] Dewanto, R., Munadi, R., and Negara, R. M. (2018). Improved load balancing on software defined network-based equal cost multipath routing in data center network. **JURNAL INFOTEL**, 10(3):157–162.

[Editor-RFC, 2019a] Editor-RFC (1989-2019a). Rfc search page. Rfc, RFC Editor. `https://www.rfc-editor.org/search/rfc_search_detail.php?pub_date_type=any` Consultado el 10/octubre/2020.

[Editor-RFC, 2019b] Editor-RFC (1989-2019b). Rfc search page. (bgp). Rfc, RFC Editor. `https://www.rfc-editor.org/search/rfc_search_detail.php?page=All&title=bgp` Consultado el 10/octubre/2020.

[Eftimie and Borcoci, 2020] Eftimie, A. and Borcoci, E. (2020). Sdn controller implementation using opendaylight: experiments. In **2020 13th International Conference on Communications (COMM)**, pages 477–481. IEEE.

[Eichenberger, 2006] Eichenberger, J. (2006). System and method for providing fast roaming. US Patent App. 11/525,431.

[Elguea and Martinez-Rios, 2017] Elguea, L. M. and Martinez-Rios, F. (2017). An efficient method to compare latencies in order to obtain the best route for SDN. **Procedia Computer Science**, 116:393 – 400. Discovery and innovation of computer science technology in artificial intelligence era: The 2nd International Conference on Computer Science and Computational Intelligence (ICCSCI 2017).

[Elguea et al., 2016] Elguea, L. M., Martinez-Rios, F. O., and Ramos, J. C. (2016). Latency analysis in IPv4/IPv6 Data Network. **International Journal of P2P Network Trends and Technology (IJPTT).**, pages V25:6–10 Apr–2016 .

[Ephremides and Verdu, 1989] Ephremides, A. and Verdu, S. (1989). Control and optimization methods in communication network problems. **IEEE Transactions on Automatic Control**, 34(9):930–942.

[Ergen and Varaiya, 2004] Ergen, M. and Varaiya, P. (2004). Throughput formulation and wlan optimization in mixed data rates for ieee 802.11 dcf mode. In **Global Telecommunications Conference Workshops, 2004. GlobeCom Workshops 2004. IEEE**, pages 266–269. IEEE.

[F5-Networks, 2012] F5-Networks (2012). Big-ip wan optimization manager. `https://www.f5.com/pdf/products/big-ip-wan-optimization-manager-ds.pdf` Consultado el 10/octubre/2020.

[Feigenbaum et al., 2005] Feigenbaum, J., Papadimitriou, C., Sami, R., and Shenker, S. (2005). A bgp-based mechanism for lowest-cost routing. **Distributed Computing**, 18(1):61–72.

[Fernández Sánchez, 2018] Fernández Sánchez, L. (2018). Sdn-sniff: monitor multi-tenant traffic in virtualized network infrastructure. B.S. thesis, Universitat Politècnica de Catalunya.

[Fuxjager et al., 2007] Fuxjager, P., Valerio, D., and Ricciato, F. (2007). The myth of non-overlapping channels: interference measurements in ieee 802.11. In **Wireless on Demand Network Systems and Services, 2007. WONS'07. Fourth Annual Conference on**, pages 1–8. IEEE.

[Gao et al., 1999] Gao, L., Zhang, Z.-L., and Towsley, D. (1999). Catching and selective catching: Efficient latency reduction techniques for delivering continuous multimedia streams. In **Proceedings of the seventh ACM international conference on Multimedia (Part 1)**, pages 203–206. ACM.

[Garg and Bonaci, 2015] Garg, P. and Bonaci, D. (2015). Virtual network routing. US Patent App. 14/026,803.

[Gero, 2020] Gero, C. E. (2020). Stream-based data deduplication using directed cyclic graphs to facilitate on-the-wire compression. US Patent App. 16/565,754.

[Gessert et al., 2020] Gessert, F., Wingerath, W., and Ritter, N. (2020). Latency in cloud-based applications. In **Fast and Scalable Cloud Data Management**, pages 13–31. Springer.

[Google, 2018] Google (2018). Estadísticas de ipv6 de google. `https://www.google.com/intl/es/ipv6/statistics.html` Consultado el 22/septiembre/2019.

[Google, 2020] Google (2020). Estadísticas de ipv6 de google. `https://www.google.com/intl/es/ipv6/statistics.html` Consultado el 28/octubre/2020.

[Gratton, 2013] Gratton, D. (2013). **The Handbook of Personal Area Networking Technologies and Protocols**. Cambridge University Press.

[Greenberg et al., 2008] Greenberg, A., Hamilton, J., Maltz, D. A., and Patel, P. (2008). The cost of a cloud: research problems in data center networks. **ACM SIGCOMM computer communication review**, 39(1):68–73. `http://research.microsoft.com/en-us/um/people/dmaltz/papers/dc-costs-ccr-editorial.pdf` Consultado el 10/octubre/2020.

[Guimaraes et al., 2020] Guimaraes, R. S., Martínez, V. M., Mello, R. C., Mafioletti, D. R., Martinello, M., and Ribeiro, M. R. (2020). An sdn-nfv orchestration for reliable and low latency mobility in off-the-shelf wifi. In **ICC 2020-2020 IEEE International Conference on Communications (ICC)**, pages 1–6. IEEE.

[Guo et al., 2015] Guo, C., Yuan, L., Xiang, D., Dang, Y., Huang, R., Maltz, D., Liu, Z., Wang, V., Pang, B., Chen, H., et al. (2015). Pingmesh: A large-scale system for data center network latency measurement and analysis. **ACM SIGCOMM Computer Communication Review**, 45(4):139–152.

[Haas and Mitchell, 2014] Haas, J. and Mitchell, J. (2014). Reservation of last autonomous system (as) numbers. **ietf org**. `https://tools.ietf.org/html/draft-ietf-idr-last-as-reservation-00` Consultado el 10/octubre/2020.

[Hamdoun et al., 2021] Hamdoun, H., Nazir, S., Alzubi, J. A., Laskot, P., and Alzubi, O. A. (2021). Performance benefits of network coding for hevc video communications in satellite networks. **Iranian Journal of Electrical and Electronic Engineering**, 17(3):1.

[Hart et al., 2020] Hart, N., Rotsos, C., Giotsas, V., Race, N., and Hutchison, D. (2020). λBGP: Rethinking BGP programmability. In **NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium**, pages 1–9.

[He et al., 2015] He, K., Khalid, J., Gember-Jacobson, A., Das, S., Prakash, C., Akella, A., Li, L. E., and Thottan, M. (2015). Measuring control plane latency in sdn-enabled switches. In **Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research**, page 25. ACM.

[Hernandez-Valencia et al., 2015] Hernandez-Valencia, E., Izzo, S., and Polonsky, B. (2015). How will nfv/sdn transform service provider opex? **IEEE Network**, 29(3):60–67.

[Hernández, 2020] Hernández, G. (2020). Aprende en casa II. `https://mexico.as.com/mexico/2020/08/25/actualidad/1598378605_735318.html`. Consultado el 9/febrero/2021.

[Heydari et al., 2019] Heydari, M., Mylonas, A., Katos, V., and Gritzalis, D. (2019). Towards indeterminacy-tolerant access control in iot. In **Handbook of Big Data and IoT Security**, pages 53–71. Springer.

[HUAWEI, 2013] HUAWEI (2013). Feature description - ip routing. `http://support.huawei.com/enterprise/documentOnline?contentId=DOC1000009646&currentPartNo=10022&togo=content`.

[Hura and Singhal, 2001] Hura, G. and Singhal, M. (2001). **Data and Computer Communications: Networking and Internetworking**. CRC Press.

[Hurricane Electric AS13679, 2020] Hurricane Electric AS13679 (2020). Hurricane electric lg bgp as13679. Updated 15 Mar 2020 21:46 PST ©2020 Hurricane Electric. `https://bgp.he.net/AS13679` Consultado el 10/octubre/2020.

[Hurricane Electric AS22566, 2020] Hurricane Electric AS22566 (2020). Hurricane electric lg bgp as22566. Updated 15 Mar 2020 21:46 PST ©2020 Hurricane Electric. `https://bgp.he.net/AS20080` Consultado el 30/octubre/2020.

[Hurricane Electric AS3649, 2020] Hurricane Electric AS3649 (2020). Hurricane electric lg bgp as3649. Updated 15 Mar 2020 21:46 PST ©2020 Hurricane Electric. `https://bgp.he.net/AS3649` Consultado el 30/octubre/2020.

[Hurricane Electric AS6503, 2020] Hurricane Electric AS6503 (2020). Hurricane electric lg bgp as6503. Updated 15 Mar 2020 21:46 PST ©2020 Hurricane Electric. `https://bgp.he.net/AS26503` Consultado el 10/octubre/2020.

[IANA, 2014] IANA (2014). Special-purpose autonomous system (as) numbers. `https://www.iana.org/assignments/iana-as-numbers-special-registry/iana-as-numbers-special-registry.xhtml` Consultado el 10/octubre/2020.

[INEE, 2019] INEE (2019). Principales cifras nacionales. Technical report. Consultado el 9/febrero/2021.

[Internap, 2016] Internap (2016). Managed internet route optimizer (miro). `http://www.internap.com/network-services/miro/` Consultado el 10/octubre/2020.

[Isto et al., 2020] Isto, P., Heikkilä, T., Mämmelä, A., Uitto, M., Seppälä, T., and Ahola, J. M. (2020). 5g based machine remote operation development utilizing digital twin. **Open Engineering**, 10(1):265–272.

[JABLONER, 2015] JABLONER, P. (2015). The two-napkin protocol. `https://www.computerhistory.org/atchm/the-two-napkin-protocol/` Consultado el 10/octubre/2020.

[Jain et al., 2013] Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., Venkata, S., Wanderer, J., Zhou, J., and Zhu, M. (2013). Experience with a globally-deployed software defined wan. **ACM SIGCOMM 2013 conference on SIGCOMM**, pages 3–14.

[Jamrozik et al., 1996] Jamrozik, H. A., Feeley, M. J., Voelker, G. M., Evans II, J., Karlin, A. R., Levy, H. M., and Vernon, M. K. (1996). Reducing network latency using subpages in a global memory environment. **ACM SIGOPS Operating Systems Review**, 30(5):258–267.

[Jan and Shieh, 2011] Jan, S.-L. and Shieh, G. (2011). Optimal sample sizes for welchs test under various allocation and cost considerations. **Behavior research methods**, 43(4):1014–1022.

[Janitor et al., 2010] Janitor, J., Jakab, F., and Kniewald, K. (2010). Visual learning tools for teaching/learning computer networks: Cisco networking academy and packet tracer. In **Networking and Services (ICNS), 2010 Sixth International Conference on**, pages 351–355. IEEE.

[Jarschel et al., 2013] Jarschel, M., Wamser, F., Hohn, T., Zinner, T., and Tran-Gia, P. (2013). Sdn-based application-aware networking on the example of youtube video streaming. **Software Defined Networks (EWSDN), 2013 Second European Workshop on**, pages 87–92.

[Jayakody et al., 2019] Jayakody, D. N. K., Srinivasan, K., and Sharma, V. (2019). **5G Enabled Secure Wireless Networks**. Springer.

[Jenkins et al., 2010] Jenkins, A., Kuzminsky, S., Gifford, K. K., Pitts, R. L., and Nichols, K. (2010). Delay/disruption-tolerant networking: flight test results from the international space station. In **Aerospace Conference, 2010 IEEE**, pages 1–8. IEEE.

[Jimenez, 1999] Jimenez, P. P. (1999). BGP implementation in java. `https://github.com/pjz/java-bgp` Consultado el 10/octubre/2020.

[Julong et al., 2019] Julong, L., Changhe, Y., Yuxiang, H., and Ziyong, L. (2019). A sdn routing optimization mechanism based on deep reinforcement learning. **Journal of Electronics and Information**, 41(11):2669–2674.

[Juniper, 2017] Juniper (2017). Bgp feature guide-juniper. download from vendor site. config-guide-routing-bgp.pdf.

[Karakaş, 2003] Karakaş, M. (2003). **Determination of network delay distribution over the internet**. PhD thesis, Citeseer.

[Keshari et al., 2020] Keshari, S. K., Kansal, V., and Kumar, S. (2020). A systematic review of quality of services (qos) in software defined networking (sdn). **Wireless Personal Communications**, pages 1–22.

[Kim et al., 2018] Kim, J., Sundaresan, K., and Kee, T. (2018). Software defined networking in a cable tv system. US Patent App. 15/992,015.

[Kouhbor et al., 2006] Kouhbor, S., Ugon, J., Mammadov, M., Rubinov, A., and Kruger, A. (2006). Coverage in wlan: Optimization model and algorithm. **algorithms**, 5:6.

[Kumar and Tech, 2014] Kumar, M. P. and Tech, B. (2014). Migration of applications from ipv4 to ipv6. **Migration**, 4(2).

[Kumar et al., 2002] Kumar, R., Parida, P. P., and Gupta, M. (2002). Topological design of communication networks using multiobjective genetic optimization. In **Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on**, volume 1, pages 425–430. IEEE.

[Lee, 1992] Lee, A. F. (1992). Optimal sample sizes determined by two–sample welch's t test. **Communications in Statistics-Simulation and Computation**, 21(3):689–696.

[Leiner et al., 2012a] Leiner, B. M., Cerf, V. G., Clark, D. D., Kahn, R. E., Kleinrock, L., Lynch, D. C., Postel, J., and Roberts, L. G. (2012a). Internet society. **A brief history of the Internet URL: http://www. internetsociety. org/internet/what-internet/history-internet/brief-history-internet [accessed 2017-07-08][WebCite Cache ID 6HxW7CLoL] Consultado el 10/octubre/2020**.

[Leiner et al., 2012b] Leiner, B. M., Cerf, V. G., Clark, D. D., Kahn, R. E., Kleinrock, L., Lynch, D. C., Postel, J., Roberts, L. G., and Wolff, S. (2012b). Brief history of the internet. **Internet Society**, 1:17.

[Letswamotse et al., 2018] Letswamotse, B. B., Malekian, R., Chen, C.-Y., and Modieginyane, K. M. (2018). Software defined wireless sensor networks (sdwsn): a review on efficient resources, applications and technologies. **Journal of Internet Technology**, 19(5):1303–1313.

[Liao et al., 2020] Liao, Z., Peng, J., Chen, Y., Zhang, J., and Wang, J. (2020). A fast q-learning based data storage optimization for low latency in data center networks. **IEEE Access**, 8:90630–90639.

[Liaskos et al., 2018] Liaskos, C., Tsioliaridou, A., Pitsillides, A., Ioannidis, S., and Akyildiz, I. (2018). Using any surface to realize a new paradigm for wireless communications. **arXiv preprint arXiv:1806.04585**.

[Lien et al., 2019] Lien, S.-Y., Tseng, C.-C., Moerman, I., and Badia, L. (2019). Recent advances in 5g technologies: New radio access and networking. **Wireless Communications and Mobile Computing**, 2019.

[Liu and Sahabi, 2020] Liu, H. A. and Sahabi, H. (2020). Traffic engineering for border gateway protocol. US Patent App. 16/194,779.

[Lorenz et al., 2017] Lorenz, C., Hock, D., Scherer, J., Durner, R., Kellerer, W., Gebert, S., Gray, N., Zinner, T., and Tran-Gia, P. (2017). An sdn/nfv-enabled enterprise network architecture offering fine-grained security policy enforcement. **IEEE communications magazine**, 55(3):217–223.

[Lougheed and Rekhter, 1989] Lougheed, K. and Rekhter, J. (1989). Border gateway protocol (bgp). RFC 1105, RFC Editor. `http://www.rfc-editor.org/rfc/rfc1105.txt` Consultado el 10/octubre/2020.

[MacDavid, 2016] MacDavid, R. C. (2016). **Compression of Interdomain SDN Policies at Exchange Points**. PhD thesis, Princeton University.

[Mace and Limited, 2014] Mace, C. and Limited, S. C. (2014). Latency on a switched ethernet network. `https://w3.siemens.com/mcms/industrial-communication/en/rugged-communication/Documents/AN8.pdf` Consultado el 21/abril/2018.

[Malik et al., 2019] Malik, A., Rashid, H. U.-R., and Azeem, W. (2019). Analysis of border gateway protocol, its types and measures to avoid risk. **LGURJCSIT**, 3(3):3–9.

[Martinez-Julia and F. Skarmeta, 2014] Martinez-Julia, P. and F. Skarmeta, A. (2014). Extending the internet of things to ipv6 with software defined networking. **www.euchina-fire.eu**.

[McMahon and Farrell, 2009] McMahon, A. and Farrell, S. (2009). Delay-and disruption-tolerant networking. **IEEE Internet Computing**, 13(6).

[Mills, 1995] Mills, A. (1995). **Understanding FDDI: A 100Mbps Solution for Today's Corporate LANs**. Prentice Hall.

[Misic and Misic, 2008] Misic, J. and Misic, V. (2008). **Wireless Personal Area Networks: Performance, Interconnection, and Security with IEEE 802.15.4**. Wireless Communications and Mobile Computing. Wiley.

[Mitchell, 2017] Mitchell, B. (2017). Top 5 network routing protocols explained @ONLINE. `https://www.lifewire.com/top-network-routing-protocols-explained-817965` Consultado el 10/octubre/2020.

[Mitchell, 2000] Mitchell, J. S. (2000). Geometric Shortest Paths and Network Optimization. **Handbook of computational geometry**, 334:633–702.

[Muluye et al., 2019] Muluye, W., Abebe, M., and Kumar, N. S. (2019). Performance analysis and evaluation of software defined networking distributed controllers in datacentre networks. **International Journal of Computational Systems Engineering**, 5(1):61–66.

[Murray et al., 2007] Murray, D., Dixon, M., and Koziniec, T. (2007). Scanning delays in 802.11 networks. In **Next Generation Mobile Applications, Services and Technologies, 2007. NGMAST'07. The 2007 International Conference on**, pages 255–260. IEEE.

[Myers and Hughes, 2020] Myers, T. J. and Hughes, M. Z. (2020). Multiple access system and method for determining a distance to an endpoint. US Patent 10,673,491.

[Nachtigall et al., 2008] Nachtigall, J., Zubow, A., and Redlich, J.-P. (2008). The impact of adjacent channel interference in multi-radio systems using ieee 802.11. In **International Wireless Communications and Mobile Computing Conference (IWCMC '08)**.

[Network Startup Resource Center, 2011] Network Startup Resource Center (2011). Prácticas recomendadas de BGP. `https://nsrc.org/workshops/2011/walc/routing/raw-attachment/wiki/Agenda/` Consultado el 10/octubre/2020.

[Nirmala, 2014] Nirmala, M. B. (2014). Wan optimization tools, techniques and research issues for cloud-based big data analytics. In **2014 World Congress on Computing and Communication Technologies**, pages 280–285. IEEE.

[Opportimes, 2020] Opportimes, R. (2020). Mercado de telecomunicaciones de méxico. `https://www.opportimes.com/televisa-tiene-12-3-del-mercado-de-telecomunicaciones-de-mexico/` Consultado el 23/enero/2021.

[Pang et al., 2019] Pang, A.-C., Au, E., Ai, B., and Zhuang, W. (2019). Guest editorial introduction to the special section on fog/edge computing for autonomous and connected cars. **IEEE Transactions on Vehicular Technology**, 68(4):3059–3060.

[Park et al., 2019] Park, Y., Hu, H., and Yuan, X. (2019). Security labs for software defined networks in cloudlab. In **Proceedings of the 50th ACM Technical Symposium on Computer Science Education**, pages 1235–1235. ACM.

[Perahia and Stacey, 2013] Perahia, E. and Stacey, R. (2013). **Next Generation Wireless LANs: 802.11n and 802.11ac**. Next Generation Wireless LANs: 802.11n, 802.11ac, and Wi-Fi Direct. Cambridge University Press.

[Philip, 2016] Philip, S. (2016). Bgp techniques for network operators. `https://www.slideshare.net/apnic/bgp-techniques-for-network-operators` Consultado el 10/octubre/2020.

[Prat, 2008] Prat, J. (2008). **Next-Generation FTTH Passive Optical Networks: Research Towards Unlimited Bandwidth Access**. Springer Netherlands.

[Proesch and Daskalaki-Proesch, 2015] Proesch, R. and Daskalaki-Proesch, A. (2015). **Technical Handbook for Radio Monitoring VHF/UHF: Edition 2013**. Books on Demand.

[Project, 2020a] Project, O. (2020a). Opendaylight ambassador program. `https://www.opendaylight.org/support/ambassadors` Consultado el 9/octubre/2020.

[Project, 2020b] Project, O. (2020b). Opendaylight project a series of lf projects, llc. `https://docs.opendaylight.org/en/latest/downloads.html` Consultado el 23/octubre/2020.

[Qazi et al., 2013] Qazi, Z. A., Tu, C.-C., Chiang, L., Miao, R., Sekar, V., and Yu, M. (2013). Simple-fying middlebox policy enforcement using sdn. **ACM SIGCOMM computer communication review**, 43(4):27–38.

[Quagga, 2002] Quagga (2002). Quagga. a routing software package. download from Kunihiro Ishiguro. quagga.pdf.

[Raps et al., 2015] Raps, Y., Kher, L., and Sharma, N. (2015). Generating optimal pathways in software-defined networking (sdn). US Patent App. 14/588,140.

[Risdianto and Mulyana, 2012] Risdianto, A. C. and Mulyana, E. (2012). Implementation and analysis of control and forwarding plane for sdn. In **Telecommunication Systems, Services, and Applications (TSSA), 2012 7th International Conference on**, pages 227–231. IEEE.

[Roch and Ashwood-Smith, 2015] Roch, E. and Ashwood-Smith, P. (2015). Compressed source routing encoding. WO Patent App. PCT/CN2015/074,052.

[Saldana et al., 2018] Saldana, J., Munilla, R., Eryigit, S., Topal, O., Ruiz-Mas, J., Fernández-Navajas, J., and Sequeira, L. (2018). Unsticking the wi-fi client: Smarter decisions using a software defined wireless solution. **IEEE Access**, 6:30917–30931.

[S.B. Mitchell, 1998] S.B. Mitchell, J. (1998). Geometric shortest path and network optimization. **Elsevier Science Publishers**, pages 633–701.

[Schlesinger et al., 2015] Schlesinger, C., Ballani, H., and Karagiannis, T. (2015). Quality of service abstractions for software-defined.

[Schreiber, 2007] Schreiber, Y. (2007). **Euclidean Shortest Paths on Polyhedra in Three Dimensions**. PhD thesis, Tel Aviv University. `http://www.cs.tau.ac.il/research/yevgeny.schreiber/SchreiberThesis.pdf` Consultado el 10/octubre/2020.

[Sezer et al., 2013] Sezer, S., Scott-Hayward, S., Chouhan, P. K., Fraser, B., Lake, D., Finnegan, J., Viljoen, N., Miller, M., and Rao, N. (2013). Are we ready for sdn? implementation challenges for software-defined networks. **IEEE Communications Magazine**, 51(7):36–43.

[Shin et al., 2012] Shin, M.-K., Nam, K.-H., and Kim, H.-J. (2012). Software-defined networking (sdn): A reference architecture and open apis. In **ICT Convergence (ICTC), 2012 International Conference on**, pages 360–361. IEEE.

[Siemens Industry, 2014] Siemens Industry, O. S. (2014). Latency on a switched ethernet network. `https://support.industry.siemens.com/cs/attachments/94772587/94772587_RUGGEDCOM_Latency_switched_Network_en.pdf` Consultado el 10/octubre/2020.

[Sminesh et al., 2020] Sminesh, C., Kanaga, E. G. M., and Sreejish, A. (2020). A multi-controller placement strategy in software defined networks using affinity propagation. **International Journal of Internet Technology and Secured Transactions**, 10(1-2):229–253.

[Song, 2013] Song, H. (2013). Protocol-oblivious forwarding: Unleash the power of sdn through a future-proof forwarding plane. In **Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking**, pages 127–132. ACM.

[Stewart III, 1998] Stewart III, J. W. (1998). **BGP4: inter-domain routing in the Internet**. Addison-Wesley Longman Publishing Co., Inc.

[Sun et al., 2019] Sun, P., Hu, Y., Lan, J., Tian, L., and Chen, M. (2019). Tide: Time-relevant deep reinforcement learning for routing optimization. **Future Generation Computer Systems**, 99:401–409.

[Sun et al., 2015] Sun, S., Gong, L., Rong, B., and Lu, K. (2015). An intelligent sdn framework for 5g heterogeneous networks. **IEEE Communications Magazine**, 53(11):142–147.

[Suresh et al., 2015] Suresh, L., Canini, M., Schmid, S., and Feldmann, A. (2015). C3: Cutting tail latency in cloud data stores via adaptive replica selection. In **Proc. 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI)**, pages 513–527, Oakland, CA. USENIX Association.

[Suzuki et al., 2014] Suzuki, K., Inagawa, S., Lippincott, J., and Cecil, A. J. (2014). Jaxa-nasa interoperability demonstration for application of dtn under simulated rain attenuation. In **AIAA 2014, Proc. SpaceOps 2014 Conf**.

[T4Labs, 2020] T4Labs (2020). Sd wan market share, market size, and industry growth drivers, 2016 - 2021. https://www.t4.ai/industry/sd-wan-market-share Consultado el 28/octubre/2020.

[Tanenbaum, 2003] Tanenbaum, A. (2003). **Redes de computadoras**. Editorial Alhambra S. A. (SP).

[Tanha, 2019] Tanha, M. (2019). **Resilient controller placement problems in software defined wide-area networks**. PhD thesis, University of Victoria.

[Teare, 2010] Teare, D. (2010). **Implementing Cisco IP routing (ROUTE) foundation learning guide: foundation learning for the ROUTE 642-902 exam**. Pearson Education.

[Tokareva et al., 2018] Tokareva, M. S., Vishnevskiy, K. O., and Chikhun, L. P. (2018). The impact of the internet of things technologies on economy. **cyberleninka.ru**, 45(3 (45) eng).

[Tomarken and Serlin, 1986] Tomarken, A. J. and Serlin, R. C. (1986). Comparison of anova alternatives under variance heterogeneity and specific noncentrality structures. **Psychological Bulletin**, 99(1):90.

[Torre, 2007] Torre, L. d. l. (2007). Intervalo de confianza para la diferencia de medias de dos distribuciones normales, varianzas desconocidas. http://www.chihuahua.tecnm.mx/academic/industrial/estadistica1/cap03d.html Consultado el 14/octubre/2020.

[Trivisonno et al., 2015] Trivisonno, R., Guerzoni, R., Vaishnavi, I., and Soldani, D. (2015). Sdn-based 5g mobile networks: architecture, functions, procedures and backward compatibility. **Transactions on Emerging Telecommunications Technologies**, 26(1):82–92.

[Uitto et al., 2019] Uitto, M., Hoppari, M., Heikkilä, T., Isto, P., Anttonen, A., and Mämmelä, A. (2019). Remote control demonstrator development in 5g test network. In **2019 European Conference on Networks and Communications (EuCNC)**, pages 101–105. IEEE.

[Vega et al., 2018] Vega, J. C., Shen, Q. C., Leon-Garcia, A., and Chow, P. (2018). Introducing recpri: A field re-configurable protocol for backhaul communication in a radio access network. **dl.ifip.org**.

[Voellmy and Wang, 2012] Voellmy, A. and Wang, J. (2012). Scalable software defined network controllers. In **Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication**, pages 289–290. ACM.

[Vukobratović and Escribano, 2020] Vukobratović, D. and Escribano, F. J. (2020). Adaptive multi-receiver coded slotted aloha for indoor optical wireless communications. **IEEE Communications Letters**.

[Wang et al., 2019] Wang, Y., Zheng, J., Tan, L., and Tian, C. (2019). Joint optimization on bandwidth allocation and route selection in qoe-aware traffic engineering. **IEEE Access**, 7:3314–3319.

[Wells et al., 1984] Wells, J., COMMAND, A., and AL., S. C. M. A. (1984). **IBM's Token-Ring LAN (Local-Area Network) – A Base-Level Communications Solution**. Defense Technical Information Center.

[Weng et al., 2018] Weng, J.-S., Weng, J., Zhang, Y., Luo, W., and Lan, W. (2018). Benbi: Scalable and dynamic access control on the northbound interface of sdn-based vanet. **IEEE Transactions on Vehicular Technology**, 68(1):822–831.

[Wirtgen et al., 2020] Wirtgen, T., De Coninck, Q., Bush, R., Vanbever, L., and Bonaventure, O. (2020). xbgp: When you can't wait for the ietf and vendors. Technical report, Université Catholique de Louvain.

[Wu et al., 2013] Wu, D. T.-S., Cho, J. S., and Ly, K. (2013). Integrating wan optimization devices with content delivery networks. US Patent 8,516,158.

[Wu and Shaofu, 2018] Wu, J. and Shaofu, Z. (2018). Software-defined data center and service cluster scheduling and traffic monitoring method therefor. US Patent App. 15/993,270.

[Wyatt et al., 2009] Wyatt, J., Burleigh, S., Jones, R., Torgerson, L., and Wissler, S. (2009). Disruption tolerant networking flight validation experiment on nasa's epoxi mission. In **Advances in Satellite and Space Communications, 2009. SPACOMM 2009. First International Conference on**, pages 187–196. IEEE.

[Yang and Cisco, 2012] Yang, Y. and Cisco (2012). Understanding switch latency. **White Paper**, 1:1–11.

[Yang et al., 2015] Yang, Y., Tsai, C., and Dong, L. (2015). Intelligent host route distribution for low latency forwarding and ubiquitous virtual machine mobility in interconnected data centers. US Patent 8,953,624 `https://www.google.com.mx/patents/US8953624` Consultado el 18/septiembre/2019.

[Yuen, 1974] Yuen, K. K. (1974). The two-sample trimmed t for unequal population variances. **Biometrika**, 61(1):165–170.

[Zhang and Bartell, 2003] Zhang, R. and Bartell, M. (2003). **BGP Design and Implementation**. Fundamentals. Pearson Education.

[Zhao et al., 2001] Zhao, X., Pei, D., Wang, L., Massey, D., Mankin, A., Wu, S. F., and Zhang, L. (2001). An analysis of bgp multiple origin as (moas) conflicts. In **Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement**, pages 31–35. ACM.

[Zhou, 2020] Zhou, Y. (2020). **Physical Layer Security and Latency Optimization for Internet of Things Communications**. PhD thesis, University of Sydney.

Apéndice A

Código fuente de los programas desarrollados

En este anexo se muestran los códigos en Java desarrollados en esta Tesis Doctoral para la realización de los experimentos y la implementación de los algoritmos de optimización de rutas.

A.1. Código Java implementación BGP

Listado A.1: BGPHeader.java

```
1  package bgp;
2
3  import java.io.*;
4
5  class BGPHeader implements Cloneable {
6
7      public byte marker[];
8      public int packetlength;
9      public static int length = 19; /* static header size */
10     public int MIN_PACKETLENGTH = length; /* smallest packet is just a header */
11     public static int MAX_PACKETLENGTH = 4096;
12     public byte type;
13     public static byte TYPE_OPEN = 1;
14     public static byte TYPE_UPDATE = 2;
15     public static byte TYPE_NOTIFICATION = 3;
16     public static byte TYPE_KEEPALIVE = 4;
17
18     public static boolean debug = false;
19
20     BGPHeader() {
21         marker = new byte[16];
22         /* init to all 1's */
23         for (int i = 0; i < marker.length; i++) {
24           marker[i] = (byte) 0xff;
25         }
26         packetlength = length;
27     }
28
29     BGPHeader(DataInputStream in) throws IOException {
30         byte headerbuffer[] = new byte[length];
31         marker = new byte[16];
32          if (debug) System.out.println(this.toString());
33         if (debug) System.out.println("HEADER: waiting for data ("+Integer.toString(length)+"
               ↪ bytes)");
34
35         in.readFully(headerbuffer, 0, length);
36         //int n = in.read(headerbuffer, 0, length); // test to see if we're getting *anything*
37         //System.out.println("HEADER: read "+Integer.toString(n)+" bytes.");
38
39         if (debug) System.out.println("HEADER: received data");
40         fromBytes(headerbuffer, 0);
41         //if (debug) System.out.println(this.toString());
42     }
43
44     public void writeTox(OutputStream out) throws IOException {
```

```
45          byte outbuffer[] = new byte[length];
46          toBytes(outbuffer, 0);
47          outbuffer[length-1]=2;
48          outbuffer[length-2]=6;
49          if (debug) System.out.println("Sending "+toString());
50          out.write(outbuffer);
51          out.flush();
52      }
53
54      public void writeTo(OutputStream out) throws IOException {
55          byte outbuffer[] = new byte[length];
56          toBytes(outbuffer, 0);
57          if (debug) System.out.println("Sending "+toString());
58          try {
59              out.write(outbuffer);
60              out.flush();
61          } catch (IOException ex) {
62              System.err.println("Error en writeTo x "+ex.toString());
63              ex.printStackTrace();
64              System.err.println("(Sending "+toString());
65          }
66      }
67
68
69      @Override
70      public final String toString() {
71          return "HEADER: length: "+Integer.toString(packetlength)+" type: "+Byte.toString(type);
72      }
73
74      public void toBytes(byte headbuffer[], int offset) {
75          System.arraycopy(marker, 0, headbuffer, offset, 16);
76          headbuffer[offset+16] = util.byteOfInt(packetlength, 1);
77          headbuffer[offset+17] = util.byteOfInt(packetlength, 0);
78          headbuffer[offset+18] = type;
79      }
80
81      public final void fromBytes(byte b[], int offset) throws IOException {
82          System.arraycopy(b, offset, marker, 0, 16);
83          packetlength = util.intFromBytes((byte) 0, (byte) 0, b[offset+16], b[offset+17]);
84          type = b[offset+18];
85      }
86
87  }
```

Listado A.2: BGPMain.java

```
1   package bgp;
2
3   import bgp.listas.profundo.Profundo;
4   import Anova.RevisaAnova;
5   import bgp.Bloquea.Seguridad;
6   import bgp.Replica.Replicador;
7   import bgp.listas.*;
8   import db.AdminDB;
9   import java.util.*;
10  import java.net.*;
11  import java.io.*;
12  import java.text.DateFormat;
13  import java.text.SimpleDateFormat;
14  import java.util.concurrent.ConcurrentHashMap;
15  import org.apache.poi.xssf.usermodel.XSSFWorkbook;
16  import weka.core.Attribute;
17  import weka.core.Instances;
18  /*
19  // 05/mar/2017
20  // Se modificó las funciones que muestran la ayuda y el resumen
21  // 06/mar/2017
22  // Se implementó en detalle de una IP (info)
23  // Se agregó Next Hop
```

```
24 // 07/mar/2017
25 // se agregó envio de paquetes...
26 // se probó en UP MIX
27 // 14/mar/2017
28 // Se agregó tabal en BD para el anuncio....
29 // 28/mar/2017
30 // se agregó HashMap para rutas
31 // y se crea respaldo para evitar exeptions...
32 // se quita respaldo y se implementa otro ciclo
33 // (http
34 ://stackoverflow.com/questions/12368012/issue-with-hashmap-getting-concurrent-modification-
       exception)
35 // se quita lo anterior y se implementa concurrentHashmap
36 //(http://stackoverflow.com/questions/26621907/how-to-avoid-hashmap-
       concurrentmodificationexception-while-manipulating-value)
37 //
38 // enero 2018  Se probó IPv6 e IPv4, hay un error no encontrado en arregla de IPv4,
       ↪ probablemente con un Nexthop que no existe.
39 // Se cambio send de NLRI para bytes...
40 // Se agrego ip y puerto de la conjexión a la identificacion (Router id) para usar ipv4 e
       ↪ ipv6 simultaneo
41 */
42 class BGPMain {
43
44     //String bind_to = "192.168.5.224"; /* what IP to bind to locally */
45     String bind_to;
46     int DEFAULT_PORT = 179;
47     /* Standard port */
48
49     public int MyASNumber = 13679;
50     //public int MyASNumber = 65534;
51     //public int MyASNumber =18734;
52     //public int MyASNumber = 21520;
53
54     /* our AS Number */
55     //public int BGP_Session_Timeout = 240;
56     public int BGP_Session_Timeout = 90;
57
58     /* 0 or >=3 ; 4 mins (240s) per RFC 1771 */
59
60     Map<String, Object> connections = new ConcurrentHashMap<>();
61
62     boolean debug = false;
63     boolean should_run = true;
64     long local_BGPID;
65     /* our BGPID */
66     ServerSocket serverport;
67     Listener listener;
68
69     ArrayList<BGPSession> sessions = new ArrayList<>();
70
71     public BGPMain() {
72         constructor(DEFAULT_PORT);
73     }
74
75     public BGPMain(int port) {
76         constructor(port);
77     }
78
79     private void constructor(int port) {
80         byte localaddr[];
81         InetAddress us;
82         try {
83             bind_to = InetAddress.getLocalHost().getHostName();
84             us = InetAddress.getByName(bind_to);
85             System.out.println("Atendiendo " + bind_to + ":" + Integer.toString(port) + "...");
86             serverport = new ServerSocket(port);
87             localaddr = us.getAddress();
88
89             local_BGPID = util.longFromBytes((byte) 0, (byte) 0, (byte) 0, (byte) 0,
90                     localaddr[0], localaddr[1], localaddr[2], localaddr[3]);
```

```
91          } catch (IOException e) {
92              System.err.println("Trouble binding to port " + Integer.toString(port) + ": " + e.
                    toString());
93              System.exit(1);
94          }
95      }
96
97
98      /* internal listening threadclass */
99      class Listener extends Thread {
100
101         BGPMain bgpl;
102         ServerSocket server;
103
104         public Listener(ServerSocket s, BGPMain b) {
105             bgpl = b;
106             server = s;
107         }
108
109         @Override
110         public void run() {
111             Socket socket;
112             BGPSession s;
113             try {
114                 while (true) {
115                     socket = server.accept();
116                     s = new BGPSession(socket, bgpl);
117                     s.start();
118                     bgpl.sessions.add(s);
119                 }
120             } catch (IOException e) {
121                 System.err.println("IO Error: " + e.toString());
122             }
123         }
124     }
125
126     /* internal threadclass to handle connections */
127     class BGPSession extends Thread {
128
129         public Socket sock;
130         BGPMain bgpl;
131
132         public BGPSession(Socket s, BGPMain b) {
133             sock = s;
134             bgpl = b;
135         }
136
137         @Override
138         public void run() {
139             bgpl.handleConnection(sock);
140         }
141
142         public boolean isIpv6() {
143             return util.esIpv6(sock.getInetAddress().toString());
144         }
145     }
146
147     public void startListening() {
148         /* spawn a new listening thread so we can get on with our life */
149         listener = new Listener(serverport, this);
150         listener.start();
151     }
152
153     public void stopListening() {
154         /* kill all child threads */
155         for (int i = 0; i < sessions.size(); i++) {
156             BGPSession session = (BGPSession) sessions.get(i);
157             session.interrupt();
158
159         }
160         //listener.stop();
161         listener.interrupt();
```

```
162
163     }
164
165     /* This will keep the other side from thinking we're dead.
166      Also, since we only listen, this is the only thread that
167      writes anything to the socket */
168     class KeepAliveThread extends Thread {
169
170         OutputStream out;
171         long holdtime;
172
173         KeepAliveThread(OutputStream who, long delay) {
174             out = who;
175             holdtime = delay;
176         }
177
178         @Override
179         public void run() {
180             BGPHeader keepalive = new BGPHeader();
181             keepalive.type = BGPHeader.TYPE_KEEPALIVE;
182             while (true) {
183                 try {
184                     KeepAliveThread.sleep(holdtime * 1000);
185                     keepalive.writeTo(out);
186                 } catch (InterruptedException e) {
187                 } catch (IOException e) {
188                     System.err.println("keepalive thread error: " + e.toString());
189                     //e.printStackTrace();
190                     System.exit(1);
191                 }
192             }
193         }
194     }
195
196     public void connectTo(String remotehost, int port) throws UnknownHostException,
        ↪ IOException {
197         Socket s = new Socket(remotehost, port);
198         BGPSession session = new BGPSession(s, this);
199         sessions.add(session);
200         session.start();
201     }
202
203     public void connectTo(String remotehost) throws UnknownHostException, IOException {
204         connectTo(remotehost, DEFAULT_PORT);
205     }
206
207     public void handleConnection(Socket s) {
208         OutputStream out;
209         DataInputStream in;
210
211         long holdtime = BGP_Session_Timeout;
212         KeepAliveThread kt = null;
213
214         try {
215             /* set up the socket i/o */
216             out = s.getOutputStream();
217             in = new DataInputStream(s.getInputStream());
218
219
220             /* STATE: ACTIVE (per BGP draft) */
221
222 /* make an OPEN packet */
223             BGPOpenPacket open = new BGPOpenPacket();
224             open.AS = MyASNumber;
225             open.version = 4;
226             /* BGP version */
227             open.BGPID = local_BGPID;
228             open.holdtime = BGP_Session_Timeout;
229             //open.params.add(bop);
230             String direccionrouter=s.getInetAddress().toString();
231             if (direccionrouter.contains(":")) {
232                 System.out.println("Conexión IPv6");
```

```
233
234                 BGPOpenParameter bop=new BGPOpenParameter();
235                 bop.type=2;
236                 bop.value=util.hexStringToByteArray("010400020001");
237                 open.params.add(bop);
238                 BGPOpenParameter bop1=new BGPOpenParameter();
239                 bop1.type=2;
240                 bop1.value=util.hexStringToByteArray("4104"+util.LengNumto4String(MyASNumber));
241                 open.params.add(bop1);
242
243             } else {
244                 System.out.println("Conexión IPv4");
245
246                 BGPOpenParameter bop=new BGPOpenParameter();
247                 bop.type=2;
248                 bop.value=util.hexStringToByteArray("010400010001");
249                 open.params.add(bop);
250                 // Al parecer, no puedo mandar solo el as de 32 bits, si no mando el
                        ↪ multiprocololo de ipv4
251                 BGPOpenParameter bop1=new BGPOpenParameter();
252                 bop1.type=2;
253                 bop1.value=util.hexStringToByteArray("4104"+util.LengNumto4String(MyASNumber));
254                 open.params.add(bop1);
255
256             }
257             /* send the OPEN packet */
258             open.writeTo(out);
259             out.flush();
260             /* STATE: OPENSENT */
261
262             /* wait for an OPEN packet */
263             open = new BGPOpenPacket(in);
264             System.out.println("Recibiendo OPEN");
265             String conexion=s.getRemoteSocketAddress().toString();
266             conexion=conexion.replaceAll("/", "");
267             System.out.println("Conectado a " + s.getLocalAddress().toString() + " desde " +
                    ↪ conexion);
268             System.out.println("AS = "+open.AS);
269             MyASNumber=open.AS;
270             /* got it, check the holdtime */
271             if (open.holdtime < holdtime) {
272                 holdtime = open.holdtime;
273             }
274
275             //System.out.println("asasasasasasas "+s.getInetAddress().toString()+" asasasasas
                    ↪ "+s.getPort() );
276
277             String nID=InetNetwork.toString(open.BGPID)+" ("+s.getInetAddress().toString()+"-
                >"+s.getPort()+")";
278             if (debug) System.out.println("NuevoID="+nID);
279
280             /* keep track of this connection (index by BGPID) */
281             //if (connections.get(InetNetwork.toString(open.BGPID)) != null) {
282             if (connections.get(nID) != null) {
283                 /* duplicate connection; no need for it, so close it */
284                 System.out.println("Conexión duplicada para BGPID " + Long.toString(open.BGPID)
                        ↪ + ", eliminando...");
285                 BGPNotificationPacket cease = new BGPNotificationPacket();
286                 cease.error = BGPNotificationPacket.CEASE;
287                 cease.writeTo(out);
288                 s.close();
289                 return;
290             } else {
291                 System.out.println("La conexión no está duplicada; Aceptando...");
292             }
293
294             BGPStatus status = new BGPStatus(open.BGPID);
295
296             connections.put(nID, status);
297
298             /* ack with a KEEPALIVE */
```

```
299             BGPHeader keepalive = new BGPHeader();
300             keepalive.type = BGPHeader.TYPE_KEEPALIVE;
301             keepalive.writeTo(out);
302             if (debug) {
303                 System.out.println("Keepalive enviado");
304             }
305
306             /* STATE: OPENCONFIRM */
307             keepalive = new BGPHeader(in);
308             if (keepalive.type == BGPHeader.TYPE_KEEPALIVE) {
309                 /* rock on */
310                 // System.out.println("received OPENCONFIRM (keepalive)");
311             } else if (keepalive.type == BGPHeader.TYPE_NOTIFICATION) {
312                 /* error probably */
313                 System.out.println("received NOTIFICATION instead of OPEN confirm");
314
315                 BGPNotificationPacket error = new BGPNotificationPacket(keepalive, in);
316                 System.out.println(error.toString());
317             } else {
318                 System.err.println("Error in FSM.  Unexpected response to OPEN.");
319             }
320
321             /* STATE: ESTABLISHED */
322             if (holdtime > 0) {
323                 /* 0 holdtime means don't send keepalives,
324         else space them < holdtime apart */
325                 if (holdtime < 3) {
326                     /* never have a keepalive of < 3s */
327                     holdtime = 3;
328                 }
329                 kt = new KeepAliveThread(out, holdtime * 9 / 10);
330                 kt.start();
331             }
332
333             boolean should_runi = true;
334             BGPHeader header;
335             while (should_runi) {
336                 // System.out.println("Waiting for header...");
337                 header = new BGPHeader(in);
338                 // System.out.print("...Got header of type: ");
339                 if (header.type == BGPHeader.TYPE_KEEPALIVE) {
340                     // System.out.println("Keepalive");
341                     /* reset the hold timer. well, but nevermind */
342                 } else if (header.type == BGPHeader.TYPE_UPDATE) {
343                     // System.out.print("Update " + Long.toString(status.updates)+" : ");
344                     status.updates++;
345                     if (status.uptime()>30000) {
346                         if (status.updates %1000==0) {
347                             //System.out.println(System.currentTimeMillis());
348                             // Tareas Programadas por actualizaciones....
349                         }
350                     }
351
352                     /* handle updates */
353                     BGPUpdatePacket update = new BGPUpdatePacket(header, in);
354                     // System.out.print(" w: "+Integer.toString(update.withdrawls.size()));
355                     for (Enumeration w = update.withdrawls.elements(); w.hasMoreElements();) {
356                         InetNetwork net = (InetNetwork) w.nextElement();
357                         // System.out.println("Withdraw: " +net.toString());
358                         status.addWithdrawl(net);
359                     }
360                     // System.out.println(" a: "+Integer.toString(update.nlrinfo.size()));
361                     //System.out.println("**************************"+update.DIPv6);
362                     if (update.DIPv6.Agregar) {
363                         for (int i=0;i<update.DIPv6.NextHop.size();i++) {
364                             for (int j=0;j<update.DIPv6.Segmentos.size();j++) {
365                                 status.addAnnounce(update.DIPv6.Segmentos.get(j),update.DIPv6.
                                     NextHop.get(i).toString(), update.Paths);
366                             }
367                         }
```

```
368                     } else {
369                         for (int j=0;j<update.DIPv6.Segmentos.size();j++) {
370                             status.addWithdrawl(update.DIPv6.Segmentos.get(j));
371                         }
372                     }
373                     for (Enumeration a = update.nlrinfo.elements(); a.hasMoreElements();) {
374                         InetNetwork net = (InetNetwork) a.nextElement();
375                         //System.out.println("Announce: "+net.toString());
376 
377                         status.addAnnounce(net, update.NextHop, update.Paths);
378                         // LME update.ProcesaAtributos();
379                     }
380                 } else if (header.type == BGPHeader.TYPE_NOTIFICATION) {
381                     System.out.println("Notification de " + InetNetwork.toString(open.BGPID)+ "
                        ↪ ("+nID+")");
382                         //keepalive = new BGPHeader(in);
383                      BGPNotificationPacket noti = new BGPNotificationPacket(header, in);
384                   System.out.println("fin x "+noti.toString());
385                         //System.exit(-1);
386                         //s.close();
387                         // Se borra la conexion...
388                         connections.remove(nID);
389                         for (int sec=0; sec<sessions.size();sec++) {
390                             BGPSession bgps=(BGPSession) sessions.get(sec);
391                             if (nID.contains(bgps.sock.getInetAddress().toString())) {
392                                 bgps.interrupt();
393                                 // y se borra la session que corresponde a esa dirección.
                                    ↪ (No se valida puerto...)
394                                 sessions.remove(sec);
395                             }
396                         }
397 
398 
399                     should_runi = false;
400                 }
401             }
402         } catch (IOException e) {
403             System.err.println("IO Error2: " + e.toString());
404             //e.printStackTrace();
405             //System.exit(-1);
406         }
407         if (kt != null) {
408             kt.interrupt();
409         }
410     }
411 
412 
413     public static void main(String[] args) {
414 
415         Properties defaultProps = System.getProperties(); //obtiene las "properties" del
            ↪ sistema
416         defaultProps.put("java.net.preferIPv6Addresses", "true");
417 
418         BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
419         BGPMain b = new BGPMain();
420 
421         b.startListening();
422 
423         String command = "";
424         while (!command.equalsIgnoreCase("quit")) {
425 
426             try {
427                 System.out.print(">");
428                 try {
429                     command = in.readLine();
430                 } catch (IOException e) {
431                     System.out.println("Invalid input");
432                 }
433                 BGPStatus status;
434                 switch (command) {
435                     case "?":
```

```
436                     case "help":
437                         Listados.Muestra_Ayuda();
438                         break;
439                     case "arregla": {
440                         try {
441                             Set<String> keys = b.connections.keySet();
442                             for (String arg : keys) {
443                                 System.out.println("Conexiones de " + arg + " is: ");
444                                 if ((status = (BGPStatus) b.connections.get(arg)) != null) {
445                                     OutputStream out=null;
446                                     // Obtengo la session para la salida de datos que
                                        ↪ coincida con ese arg.
447                                     for (int sesid = 0; sesid < b.sessions.size(); sesid++) {
448                                         BGPSession bses = (BGPSession) b.sessions.get(sesid);
449                                         if (arg.contains(bses.sock.getInetAddress().
                                            toString())) {
450                                             out = bses.sock.getOutputStream();
451                                         }
452                                     }
453
454                                     if (util.esIpv6( arg)) {
455                                         // No se considera Arregando IPv6
456                                         ArrayList<BGPSendUpdatePacket> bu = Listados.
                                            Arregla6(status.routes);
457                                         System.out.println("Se tienen "+bu.size()+" anuncios");
458                                         for (int anuncio = 0; anuncio < bu.size(); anuncio++)
                                            ↪ {
459
460                                             BGPSendUpdatePacket bgpsu = bu.get(anuncio);
461                                             System.out.println("nh="+bgpsu.Path_NextHopIP+"
                                                ↪ pr="+bgpsu.NLRI_PrefijoRed+" R IP="+bgpsu.
                                                NLRI_RedIP);
462
463                                             // Aqui no se modifican W..
464                                             System.out.println("Areglando " + (anuncio + 1) +
                                                ↪ "/" + bu.size() + ".- " + bgpsu.NLRI_RedIP +
                                                ↪ "/" + bgpsu.NLRI_PrefijoRed);
465                                             bgpsu.writeTo6(out);
466                                             //out.flush();
467                                         }
468                                     } else {
469                                         // Arregando IPv4
470                                         ArrayList<BGPSendUpdatePacket> bu = Listados.
                                            Arregla4(status.routes);
471
472                                         for (int anuncio = 0; anuncio < bu.size(); anuncio++)
                                            ↪ {
473
474                                             BGPSendUpdatePacket bgpsu = bu.get(anuncio);
475                                             // Aqui no se modifican W..
476                                             System.out.println("Arreglando " + (anuncio + 1) +
                                                ↪ "/" + bu.size() + ".- " + bgpsu.NLRI_RedIP +
                                                ↪ "/" + bgpsu.NLRI_PrefijoRed);
477                                             bgpsu.writeTo(out);
478                                             //out.flush();
479                                         }
480                                     }
481                                 }
482                             }
483                         } catch (IOException e) {
484                             System.err.println("a "+e.toString());
485                         }
486                     }
487                     break;
488                     case "bloquea": {
489                         try {
490                             Set<String> keys = b.connections.keySet();
491                             for (String arg : keys) {
492                                 System.out.println("Conexiones de " + arg + " is: ");
493                                 if ((status = (BGPStatus) b.connections.get(arg)) != null) {
```

```
494                             OutputStream out=null;
495                             // Obtengo la session para la salida de datos que
                                    ↪ coincida con ese arg.
496                             for (int sesid = 0; sesid < b.sessions.size(); sesid++) {
497                                 BGPSession bses = (BGPSession) b.sessions.get(sesid);
498                                 if (arg.contains(bses.sock.getInetAddress().
                                        toString())) {
499                                     out = bses.sock.getOutputStream();
500                                 }
501                             }
502
503                             if (util.esIpv6( arg)) {
504                                 // Seguridad IPv6
505                                 ArrayList<BGPSendUpdatePacket> bu = Seguridad.
                                        Bloquea6(status.routes);
506                                 System.out.println("Se tienen "+bu.size()+" anuncios");
507                                 for (int anuncio = 0; anuncio < bu.size(); anuncio++)
                                        ↪ {
508
509                                     BGPSendUpdatePacket bgpsu = bu.get(anuncio);
510                                     System.out.println("nh="+bgpsu.Path_NextHopIP+"
                                            ↪ pr="+bgpsu.NLRI_PrefijoRed+" R IP="+bgpsu.
                                            NLRI_RedIP);
511
512                                     // Aqui no se modifican W..
513                                     System.out.println("Bloqueando " + (anuncio + 1) +
                                            ↪ "/" + bu.size() + ".- " + bgpsu.NLRI_RedIP +
                                            ↪ "/" + bgpsu.NLRI_PrefijoRed);
514                                     bgpsu.writeTo6(out);
515                                     //out.flush();
516                                 }
517
518                             } else {
519                                 // Seguridad IPv4
520                                 ArrayList<BGPSendUpdatePacket> bu = Seguridad.
                                        Bloquea4(status.routes);
521
522                                 for (int anuncio = 0; anuncio < bu.size(); anuncio++)
                                        ↪ {
523
524                                     BGPSendUpdatePacket bgpsu = bu.get(anuncio);
525                                     // Aqui no se modifican W..
526                                     System.out.println("Bloqueando " + (anuncio + 1) +
                                            ↪ "/" + bu.size() + ".- " + bgpsu.NLRI_RedIP +
                                            ↪ "/" + bgpsu.NLRI_PrefijoRed);
527                                     bgpsu.writeTo(out);
528                                     //out.flush();
529                                 }
530                             }
531                         }
532                     }
533                 } catch (IOException e) {
534                     System.err.println("a "+e.toString());
535                 }
536             }
537             break;
538             case "arregla4": {
539                 System.out.println("Procesando solo IPv4");
540                 try {
541                     Set<String> keys = b.connections.keySet();
542                     for (String arg : keys) {
543                         if ((status = (BGPStatus) b.connections.get(arg)) != null) {
544                             OutputStream out=null;
545                             // Obtengo la session para la salida de datos que
                                    ↪ coincida con ese arg.
546                             for (int sesid = 0; sesid < b.sessions.size(); sesid++) {
547                                 BGPSession bses = (BGPSession) b.sessions.get(sesid);
548                                 if (arg.contains(bses.sock.getInetAddress().
                                        toString())) {
549                                     out = bses.sock.getOutputStream();
```

```
550                                 }
551                             }
552
553                             if (!util.esIpv6( arg)) {
554                                 System.out.println("Conexiones de " + arg + " is: ");
555                                 // Arregando IPv4
556                                 ArrayList<BGPSendUpdatePacket> bu = Listados.
                                        Arregla4(status.routes);
557
558                                 for (int anuncio = 0; anuncio < bu.size(); anuncio++)
                                        ↪ {
559
560                                     BGPSendUpdatePacket bgpsu = bu.get(anuncio);
561                                     // Aqui no se modifican W..
562                                     System.out.println("Enviando " + (anuncio + 1) +
                                            ↪ "/" + bu.size() + ".- " + bgpsu.NLRI_RedIP +
                                            ↪ "/" + bgpsu.NLRI_PrefijoRed);
563                                     bgpsu.writeTo(out);
564                                     //out.flush();
565                                 }
566                             }
567                         }
568                     }
569
570                 } catch (IOException e) {
571                     System.err.println("a "+e.toString());
572                 }
573             }
574             break;
575             case "arregla6": {
576                 System.out.println("Procesando solo IPv6");
577                 try {
578                     Set<String> keys = b.connections.keySet();
579                     for (String arg : keys) {
580                         if ((status = (BGPStatus) b.connections.get(arg)) != null) {
581                             OutputStream out=null;
582                             // Obtengo la session para la salida de datos que
                                    ↪ coincida con ese arg.
583                             for (int sesid = 0; sesid < b.sessions.size(); sesid++) {
584                                 BGPSession bses = (BGPSession) b.sessions.get(sesid);
585                                 if (arg.contains(bses.sock.getInetAddress().
                                        toString())) {
586                                     out = bses.sock.getOutputStream();
587                                 }
588                             }
589
590                             if (util.esIpv6( arg)) {
591                                 System.out.println("Conexiones de " + arg + " is: ");
592                                 // Arregando IPv4
593                                 ArrayList<BGPSendUpdatePacket> bu = Listados.
                                        Arregla6(status.routes);
594
595                                 for (int anuncio = 0; anuncio < bu.size(); anuncio++)
                                        ↪ {
596
597                                     BGPSendUpdatePacket bgpsu = bu.get(anuncio);
598                                     // Aqui no se modifican W..
599                                     System.out.println("Enviando " + (anuncio + 1) +
                                            ↪ "/" + bu.size() + ".- " + bgpsu.NLRI_RedIP +
                                            ↪ "/" + bgpsu.NLRI_PrefijoRed);
600                                     bgpsu.writeTo6(out);
601                                     //out.flush();
602                                 }
603                             }
604                         }
605                     }
606
607                 } catch (IOException e) {
608                     System.err.println("a "+e.toString());
609                 }
```

```
610                     }
611                     break;
612                     case "test6": {
613                         System.out.println("Procesando solo IPv6");
614                         try {
615                             Set<String> keys = b.connections.keySet();
616                             for (String arg : keys) {
617                                 if ((status = (BGPStatus) b.connections.get(arg)) != null) {
618                                     OutputStream out=null;
619                                     // Obtengo la session para la salida de datos que
                                        ↪ coincida con ese arg.
620                                     for (int sesid = 0; sesid < b.sessions.size(); sesid++) {
621                                         BGPSession bses = (BGPSession) b.sessions.get(sesid);
622                                         if (arg.contains(bses.sock.getInetAddress().
                                            toString())) {
623                                             out = bses.sock.getOutputStream();
624                                         }
625                                     }
626
627                                     if (util.esIpv6( arg)) {
628                                         System.out.println("Conexiones de " + arg + " is: ");
629                                         // Arregando IPv4
630                                         ArrayList<BGPSendUpdatePacket> bu = Listados.
                                            Arregla6x(status.routes);
631
632                                         for (int anuncio = 0; anuncio < bu.size(); anuncio++)
                                            ↪ {
633
634                                             BGPSendUpdatePacket bgpsu = bu.get(anuncio);
635                                             // Aqui no se modifican W..
636                                             System.out.println("Enviando " + (anuncio + 1) +
                                                ↪ "/" + bu.size() + ".- " + bgpsu.NLRI_RedIP +
                                                ↪ "/" + bgpsu.NLRI_PrefijoRed);
637                                             bgpsu.writeTo6(out);
638                                             //out.flush();
639                                         }
640                                     }
641                                 }
642                             }
643
644                         } catch (IOException e) {
645                             System.err.println("a "+e.toString());
646                         }
647                     }
648                     break;
649                     case "arreglabd": {
650                         System.out.println("Procesando con BD");
651                         try {
652                             AdminDB bd= new AdminDB();
653                             //bd.conecta();
654                             System.out.println("Conectado");
655                             Set<String> keys = b.connections.keySet();
656                             for (String arg : keys) {
657                                 if ((BGPStatus) b.connections.get(arg) != null) {
658                                     OutputStream out=null;
659                                     // Obtengo la session para la salida de datos que
                                        ↪ coincida con ese arg.
660                                     for (int sesid = 0; sesid < b.sessions.size(); sesid++) {
661                                         BGPSession bses = (BGPSession) b.sessions.get(sesid);
662                                         if (arg.contains(bses.sock.getInetAddress().
                                            toString())) {
663                                             out = bses.sock.getOutputStream();
664                                         }
665                                     }
666
667                                     System.out.println("Conexiones de " + arg + " is: ");
668
669                                     //ArrayList<BGPSendUpdatePacket> bu = Listados.
                                        Arregla4(status.routes);
670                                     bd.conecta();
```

```
671                             ArrayList<BGPSendUpdatePacket> bu = bd.
                                    LecturaArreglaBD(arg);
672
673                             for (int anuncio = 0; anuncio < bu.size(); anuncio++) {
674
675                                 BGPSendUpdatePacket bgpsu = bu.get(anuncio);
676                                 // Aqui no se modifican W..
677                                 System.out.println("Enviando " + (anuncio + 1) + "/" +
                                        ↪ bu.size() + ".- " + bgpsu.NLRI_RedIP + "/" +
                                        ↪ bgpsu.NLRI_PrefijoRed);
678                                 if (util.esIpv6(bgpsu.Path_NextHopIP.toString())) {
679                                     bgpsu.writeTo6(out);
680                                     //out.flush();
681                                 } else {
682                                     bgpsu.writeTo(out);
683                                     //out.flush();
684
685                                 }
686                             }
687
688                         }
689                     }
690                     bd.cierra();
691                 } catch (IOException e) {
692                     System.err.println("a "+e.toString());
693                 }
694             }
695             break;
696             case "replica": {
697                 System.out.println("Replicando 100 segmentos");
698                 iniciaReplica(b.sessions, b.connections);
699             }
700             break;
701             case "hongo": {
702                 for (int sesid = 0; sesid < b.sessions.size(); sesid++) {
703                     BGPSession bses = (BGPSession) b.sessions.get(sesid);
704                     System.out.println("Session " + (sesid + 1) + ".- " + bses.bgpl.bind_
                            to);
705                     System.out.println("Session " + (sesid + 1) + ".- " + bses.bgpl.
                            toString());
706                     System.out.println("Session " + (sesid + 1) + ".- " + bses.
                            getName());
707                     System.out.println("Session " + (sesid + 1) + ".- " + bses.sock.
                            getInetAddress());
708
709
710                 }
711
712                 Set<String> keys = b.connections.keySet();
713                 for (String arg : keys) {
714                     System.out.println("Conexiones de " + arg + " is: ");
715                     if ((status = (BGPStatus) b.connections.get(arg)) != null) {
716                         System.out.println("Conexiones de " + arg + " Activa");
717                     }
718                 }
719             }
720             break;
721             case "anova": {
722                 //BGPUpdatePacket update;
723                 System.out.println("**** Este Modulo solo revisa redes IPv4 ****");
724                 Set<String> keys = b.connections.keySet();
725                 for (String arg : keys) {
726                     if (!arg.contains(":")) {
727                         System.out.println("Conexiones de " + arg + " son: ");
728                         if ((status = (BGPStatus) b.connections.get(arg)) != null) {
729                             //Map<String,Ruta> respa=status.routes;
730                             RevisaAnova.RevisaN(status.routes);
731                         }
732                     }
733                 }
```

```
734                         }
735                             break;
736                         case "aprende":
737                             //DateFormat df = new SimpleDateFormat("yyyyMMdd_HHmm");
738                             //String archivoTP="reporte"+df.format(Calendar.getInstance().
                                    getTime())+".txt";
739                             String archivoTP="Mapa.hytm.xtm";
740                             System.out.println("Generando Mapa Tematico: "+archivoTP);
741                             //PrintStream o = new PrintStream(new File("reporte.txt"));
742                             PrintStream otp = new PrintStream(new File(archivoTP));
743                             // Store current System.out before assigning a new value
744                             PrintStream consoletp = System.out;
745                             // Assign o to output stream
746                             System.setOut(otp);
747                             System.out.println(bgp.listas.TopicMaps.Genera.Inicio());
748
749                             Set<String> llaveA = b.connections.keySet();
750                             for (String arg : llaveA) {
751                                 System.out.println("<!-- Conexiones de " + arg + " son: -->");
752                                 String Familia="-";
753                                 if ((status = (BGPStatus) b.connections.get(arg)) != null) {
754                                     //Map<String,Ruta> respa=status.routes;
755                                     if (arg.contains(":")) {
756                                         Familia="ipv6";
757                                     } else {
758                                         Familia="ipv4";
759                                     }
760                                     Aprende.Procesa(status.routes, Familia);
761                                 }
762                             }
763                             System.out.println(bgp.listas.TopicMaps.Genera.Fin());
764                             otp.close();
765                             System.setOut(consoletp);
766                             System.out.println("Fin del Mapa tematico");
767                             break;
768                         case "xls":
769                         {
770                             String Archivo="Exporta.xlsx";
771                             XSSFWorkbook workbook = new XSSFWorkbook();
772
773                             System.out.println("Exportando a "+Archivo);
774
775                             Set<String> llaveXLS = b.connections.keySet();
776                             for (String arg : llaveXLS) {
777                                 System.out.println("<!-- Conexiones de " + arg + " son: -->");
778                                 String Familia="-";
779                                 if ((status = (BGPStatus) b.connections.get(arg)) != null) {
780                                     //Map<String,Ruta> respa=status.routes;
781                                     if (arg.contains(":")) {
782                                         Familia="ipv6";
783                                     } else {
784                                         Familia="ipv4";
785                                     }
786                                     Excel.Procesa(status.routes, Familia, workbook);
787                                 }
788                             }
789                             System.out.println("Guardando...");
790                             long tiempo=System.currentTimeMillis();
791                             try (FileOutputStream outputStream = new FileOutputStream(Archivo)) {
792                             workbook.write(outputStream);
793                             } catch (Exception ex) {
794                                 System.err.println(ex.toString());
795                             }
796                             System.out.println("Tardo "+(System.currentTimeMillis()-tiempo)+"ms.");
797                             System.out.println("Fin del XLS");
798                         }
799                             break;
800                         case "arff":
801                         {
802                             String Archivo="Exporta.arff";
803                             ArrayList       atts;
```

```
804                     ArrayList       Proveedores;
805                     ArrayList       StatusPrepend;
806                     Instances        data;
807                     double[]         vals;
808                     int              i;
809
810
811                     System.out.println("Exportando a "+Archivo);
812                     atts = new ArrayList();
813
814                     atts.add(new Attribute("Segmento", (ArrayList) null));  // Cadena
815                     atts.add(new Attribute("Tama_Segmento"));  // Numerico
816                     atts.add(new Attribute("Camino", (ArrayList) null));    // Cadena
817                     atts.add(new Attribute("Camino_SinPrepend", (ArrayList) null));    //
                            ↪ Cadena
818                     atts.add(new Attribute("Saltos"));                      // Numerico
819                     atts.add(new Attribute("Saltos_SinPrepend"));                       //
                            ↪ Numerico
820                     // Next-Hops
821                     Proveedores = new ArrayList();
822                     Set<String> llavet = b.connections.keySet();
823                     for (String arg : llavet) {
824                         if ((status = (BGPStatus) b.connections.get(arg)) != null) {
825                             ArrayList vdindiv=VecinosDirectos.Procesa(status.routes);
826                             for (int vd=0;vd<vdindiv.size();vd++) {
827                                 if (!Proveedores.contains(vdindiv.get(vd))) {
828                                     Proveedores.add(vdindiv.get(vd));
829                                     System.out.println("Agregando a "+vdindiv.get(vd));
830                                 }
831                             }
832                         }
833                     }
834
835                     atts.add(new Attribute("Vecino", Proveedores));             // Nominal
836
837                     StatusPrepend = new ArrayList();
838                         StatusPrepend.add("Si Prepend");
839                         StatusPrepend.add("No Prepend");
840                     atts.add(new Attribute("Prepend", StatusPrepend));
841
842                     ArrayList Protocolo = new ArrayList();
843                         Protocolo.add("ipv4");
844                         Protocolo.add("ipv6");
845
846                     atts.add(new Attribute("Protocolo", Protocolo));  // Cadena
847
848                     data = new Instances("Rutas_BGP", atts, 0);
849                     Set<String> llave = b.connections.keySet();
850                     for (String arg : llave) {
851                         System.out.println("<!-- Conexiones de " + arg + " son: -->");
852                         String Familia="-";
853                         if ((status = (BGPStatus) b.connections.get(arg)) != null) {
854                             //Map<String,Ruta> respa=status.routes;
855                             if (arg.contains(":")) {
856                                 Familia="ipv6";
857                             } else {
858                                 Familia="ipv4";
859                             }
860                             Arff.Procesa(status.routes, Familia, data, Proveedores);
861                         }
862                     }
863                     System.out.println("Guardando...");
864                     long tiempo=System.currentTimeMillis();
865                     try {
866                         BufferedWriter writer = new BufferedWriter(new
                                ↪ FileWriter(Archivo));
867                         writer.write(data.toString());
868                         writer.flush();
869                         writer.close();
870                     } catch (Exception e) {
871                         System.err.println(e.toString());
```

```
872                             e.printStackTrace();
873                         }
874                         System.out.println("Tardo "+(System.currentTimeMillis()-tiempo)+"ms.");
875                         System.out.println("Fin del ARFF");
876                     }
877                         break;
878 
879 
880                     case "profundo":
881                      try {
882                         System.out.println("Procesando TODO");
883                         long tiempop=System.currentTimeMillis();
884                         Set<String> llaveProf = b.connections.keySet();
885                         for (String arg : llaveProf) {
886                             System.out.println("<!-- Conexiones de " + arg + " son: -->");
887                             String Familia="-";
888                             if ((status = (BGPStatus) b.connections.get(arg)) != null) {
889                                 //Map<String,Ruta> respa=status.routes;
890                                 if (arg.contains(":")) {
891                                     Familia="ipv6";
892                                 } else {
893                                     Familia="ipv4";
894                                 }
895                                 Profundo.Procesa(status.routes, Familia);
896                             }
897                         }
898                         System.out.println("Tardo "+(System.currentTimeMillis()-tiempop)+"ms.");
899                     } catch (Exception e) {
900                         //System.out.println(e.toString());
901                     }
902                     break;
903                     case "revisa":
904                         //BGPUpdatePacket update;
905                         Set<String> keys = b.connections.keySet();
906                         for (String arg : keys) {
907                             System.out.println("Conexiones de " + arg + " son: ");
908                             if ((status = (BGPStatus) b.connections.get(arg)) != null) {
909                                 //Map<String,Ruta> respa=status.routes;
910                                 RevisaConsola.RevisaN(status.routes);
911                             }
912                         }
913                         break;
914                     case "paises":
915                         //BGPUpdatePacket update;
916                         Set<String> llave = b.connections.keySet();
917                         for (String arg : llave) {
918                             System.out.println("Conexiones de " + arg + " son: ");
919                             if ((status = (BGPStatus) b.connections.get(arg)) != null) {
920                                 //Map<String,Ruta> respa=status.routes;
921                                 CambioPais.Revisa(status.routes);
922                             }
923                         }
924                         break;
925                     case "estadistica":
926                         //BGPUpdatePacket update;
927                         Set<String> llave2 = b.connections.keySet();
928                         for (String arg : llave2) {
929                             System.out.println("Conexiones de " + arg + " son: ");
930                             if ((status = (BGPStatus) b.connections.get(arg)) != null) {
931                                 //Map<String,Ruta> respa=status.routes;
932                                 EstadisticasPaises.Revisa(status.routes);
933                             }
934                         }
935                         break;
936                     case "exporta":
937                         //BGPUpdatePacket update;
938                         Set<String> llavee = b.connections.keySet();
939                         for (String arg : llavee) {
940                             System.out.println("Conexiones de " + arg + " son: ");
941                             if ((status = (BGPStatus) b.connections.get(arg)) != null) {
```

```
942                         //Map<String,Ruta> respa=status.routes;
943                         ExportaDatos.Revisa(status.routes);
944                     }
945                 }
946                 break;
947             case "procesa":
948                 //BGPUpdatePacket update;
949                 Set<String> llavep = b.connections.keySet();
950                 for (String arg : llavep) {
951                     System.out.println("Conexiones de " + arg + " son: ");
952                     if ((status = (BGPStatus) b.connections.get(arg)) != null) {
953                         //Map<String,Ruta> respa=status.routes;
954                         Procesa_AS.Revisa(status.routes);
955                     }
956                 }
957                 break;
958             case "reporte":
959                 DateFormat df = new SimpleDateFormat("yyyyMMdd_HHmm");
960                 String archivo="reporte"+df.format(Calendar.getInstance().getTime())+".
                        txt";
961                 System.out.println("Generando Reporte: "+archivo);
962                 //PrintStream o = new PrintStream(new File("reporte.txt"));
963                 PrintStream o = new PrintStream(new File(archivo));
964                 // Store current System.out before assigning a new value
965                 PrintStream console = System.out;
966                 // Assign o to output stream
967                 System.setOut(o);
968                 Set<String> llaves = b.connections.keySet();
969                 for (String arg : llaves) {
970                     System.out.println("\\chapter{Conexiones de " + arg + "}\n");
971                     SimpleDateFormat dt1 = new SimpleDateFormat("dd 'de' MMMM 'de'
                            ↪ yyyy 'a' hh:mm:ss a");
972                     System.out.println("Reporte de conexiones el "+dt1.format(new
                            ↪ Date()));
973                     if ((status = (BGPStatus) b.connections.get(arg)) != null) {
974                         Reporte.reporteLatex(status.routes);
975                     }
976                 }
977                 o.close();
978                 System.setOut(console);
979                 System.out.println("Fin del Reporte");
980                 break;
981             case "grafo":
982                 System.out.println("Generando Grapho");
983                 Set<String> llavesg = b.connections.keySet();
984                 for (String arg : llavesg) {
985                     //System.out.println("\\chapter{Conexiones de " + arg + "}\n");
986                     //SimpleDateFormat dt1 = new SimpleDateFormat("dd 'de' MMMM 'de'
                            ↪ yyyy 'a' hh:mm:ss a");
987                     //System.out.println("Reporte de conexiones el "+dt1.format(new
                            ↪ Date()));
988                     if ((status = (BGPStatus) b.connections.get(arg)) != null) {
989                         ReporteGraphos.Reporte(status.routes);
990                     }
991                 }
992                 System.out.println("Fin del Grapho");
993                 break;
994             case "i":
995             case "info":
996                 //BGPUpdatePacket update;
997                 Set<String> keys1 = b.connections.keySet();
998                 for (String arg : keys1) {
999                     System.out.println("Conexiones de " + arg + " is: ");
1000                    if ((status = (BGPStatus) b.connections.get(arg)) != null) {
1001                        Listados.Detalle(status.routes);
1002                    }
1003                }
1004                break;
1005            case "m":
1006            case "mex":
```

```
1007                        //BGPUpdatePacket update;
1008                        Set<String> keys2 = b.connections.keySet();
1009                        for (String arg : keys2) {
1010                            System.out.println("Conexiones de " + arg + " is: ");
1011                            if ((status = (BGPStatus) b.connections.get(arg)) != null) {
1012                                Listados.DetalleMexico(status.routes);
1013                            }
1014                        }
1015                        break;
1016                    case "status":
1017                        break;
1018                    case "st":
1019                        break;
1020                }
1021                if (command.startsWith("connect")) {
1022                    int i, j;
1023                    String arg;
1024
1025                    for (i = 7; i < command.length() && Character.isWhitespace(command.
                            charAt(i)); i++) {
1026                        // Ciclo nada mas para contar los espacios (no fue mi idea)
1027                    }
1028                    /* LME Revisar que onda  */
1029
1030                    if (i < command.length()) {
1031                        /* there's another word, it's probably a connection */
1032                        for (j = i; j < command.length() && !Character.isWhitespace(command.
                                charAt(j)); j++) {
1033                            // Ciclo nada mas para contar los espacios (no fue mi idea)
1034                        }
1035                        arg = command.substring(i, j);
1036                        try {
1037                            b.connectTo(arg);
1038                            //b.connectTo(arg, 180);
1039                        } catch (UnknownHostException e) {
1040                            System.out.println("Unknown host: " + arg);
1041                        } catch (IOException e) {
1042                            System.out.println("Problem connecting to " + arg + ": " + e.
                                    toString());
1043                        }
1044                    }
1045                } else if (command.startsWith("status") || command.startsWith("st")) {
1046                    int i, j;
1047                    String arg;
1048                    // see if there's another word
1049                    for (i = 6; i < command.length() && Character.isWhitespace(command.
                            charAt(i)); i++) {
1050                        // Ciclo nada mas para contar los espacios (no fue mi idea)
1051                    }
1052                    if (i < command.length()) {
1053                        /* there's another word, it's probably a connection */
1054                        // get the 2nd word
1055                        for (j = i; j < command.length() && !Character.isWhitespace(command.
                                charAt(j)); j++) {
1056                            // Ciclo nada mas para contar los espacios (no fue mi idea)
1057                        }
1058                        arg = command.substring(i, j);
1059                        // the 2nd word is a connection, display its routes
1060                        if ((status = (BGPStatus) b.connections.get(arg)) != null) {
1061                            //for(Enumeration e = status.routes.keys(); e.hasMoreElements();) {
1062                            //    String key = (String)e.nextElement();
1063                            //    System.out.println(" "+key);
1064                            //}
1065                            //for (int indi=0;indi<status.routes.size();indi++) {
1066                            //    System.out.println(status.routes.get(indi).IP);
1067                            //}
1068                            Set<String> llaves = status.routes.keySet();
1069                            for (String key : llaves) {
1070                                System.out.println(status.routes.get(key)+" s="+status.
                                        toString());
```

```
1071                            }
1072                        } else {
1073                            System.out.println("Invalid argument: " + arg);
1074                        }
1075                    } else // no 2nd word, display list of connections
1076                        if (b.connections.size() < 1) {
1077                            System.out.println("No connections to list.");
1078                        } else {
1079
1080                            Listados.Muestra_Stat_Resumen(b.connections);
1081                        }
1082                }
1083            } catch (FileNotFoundException ex) {
1084                System.err.println(ex.toString());
1085            }
1086        }
1087
1088        BGPNotificationPacket cease = new BGPNotificationPacket();
1089        cease.error = BGPNotificationPacket.CEASE;
1090        cease.subcode = 2;
1091
1092
1093        for (int sesid = 0; sesid < b.sessions.size(); sesid++) {
1094            BGPSession bses = (BGPSession) b.sessions.get(sesid);
1095            System.out.println("Session " + (sesid + 1) + ".- " + bses.bgpl.bind_to+" =>"+bses.
                sock.getInetAddress()+":"+bses.sock.getRemoteSocketAddress());
1096            OutputStream out;
1097            try {
1098                out = bses.sock.getOutputStream();
1099                cease.manda(out);
1100                //out.flush();
1101            } catch (IOException e) {
1102                System.err.println(e.toString());
1103            }
1104
1105        }
1106
1107        System.out.println("Quiting...");
1108        b.stopListening();
1109        System.out.println("Quiting OK");
1110        System.exit(0);
1111    }
1112
1113    public static void iniciaReplica(ArrayList<BGPSession> sessiones, Map<String, Object>
        ↪ conexiones) {
1114
1115        BGPStatus status;
1116
1117        //for (int sesid = 0; sesid < sessiones.size(); sesid++) {
1118        //    BGPSession bses = (BGPSession) sessiones.get(sesid);
1119        //    System.out.println("Session " + (sesid + 1) + ".- " + bses.sock.getInetAddress()+"
            ↪ ipv6:"+bses.isIpv6());
1120        //}
1121
1122        try {
1123            Set<String> keys = conexiones.keySet();
1124            /*
1125            for (String arg : keys) {
1126                if ((status = (BGPStatus) conexiones.get(arg)) != null) {
1127                    System.out.println("Conexiones de " + arg + " Activa");
1128                    for (int sesid = 0; sesid < sessiones.size(); sesid++) {
1129                        BGPSession bses = (BGPSession) sessiones.get(sesid);
1130                        //System.out.println("Session " + (sesid + 1) + ".- " + bses.sock.
                            getInetAddress()+" ipv6:"+bses.isIpv6());
1131                        if (arg.contains(bses.sock.getInetAddress().toString())) {
1132                            System.out.println("Session " + (sesid + 1) + ".- " + bses.sock.
                                getInetAddress()+" ipv6:"+bses.isIpv6());
1133                        }
1134                    }
1135                }
1136            }
```

```
1137                 */
1138
1139                 System.out.println("Replicando...");
1140                 //String ini="/2801:f0:20:880:0:0:0:b";
1141                 //String fin="/2801:f0:20:c20:172:25:150:116";
1142                 //fin="/2801:f0:20:4000:0:0:0:2";
1143
1144                 String ini="/192.100.201.202";
1145                 String fin="/172.25.150.116";
1146
1147                 Map<String,Ruta> Origen=null;
1148                 OutputStream salida=null;
1149
1150                 boolean fuente_origen=false;
1151                 boolean destino_final=false;
1152
1153                 for (String arg : keys) {
1154                     if ((status = (BGPStatus) conexiones.get(arg)) != null) {
1155                         //System.out.println("Buscando entrada="+ini+" con "+arg);
1156                         if (arg.contains(ini+"-")) {
1157                             Origen=status.routes;
1158                             System.out.println("Encontrado Origen en "+arg);
1159                             fuente_origen=true;
1160                         }
1161                     }
1162                 }
1163                 for (int sesid = 0; sesid < sessiones.size(); sesid++) {
1164                     BGPSession bses = sessiones.get(sesid);
1165                     //System.out.println("Buscando Salida="+fin+" con "+bses.sock.getInetAddress().
                         toString());
1166                     if (bses.sock.getInetAddress().toString().equalsIgnoreCase(fin)) {
1167                         salida = bses.sock.getOutputStream();
1168                         System.out.println("Encontrado Destino en "+bses.sock.getInetAddress().
                             toString());
1169                         destino_final=true;
1170                     }
1171                 }
1172                 if (fuente_origen && destino_final) {
1173                     Replicador.replica(salida, Origen);
1174                 } else {
1175                     System.out.println("No se encontro Origen y/o Destino");
1176                 }
1177             } catch (IOException e) {
1178                 System.err.println(e.toString());
1179             }
1180         }
1181
1182 }
```

Listado A.3: BGPNotificationPacket.java

```
1  package bgp;
2
3  import java.io.*;
4
5  class BGPNotificationPacket {
6
7      public BGPHeader header;
8
9      public int error;
10     public static int MESSAGE_HEADER_ERROR = 1;
11     public static int OPEN_MESSAGE_ERROR = 2;
12     public static int UPDATE_MESSAGE_ERROR = 3;
13     public static int HOLD_TIMER_EXPIRED = 4;
14     public static int FINITE_STATE_MACHINE_ERROR = 5;
15     public static int CEASE = 6;
16
17     public int subcode;
18     public byte[] data;
```

```
19
20      public BGPNotificationPacket() {
21          header = new BGPHeader();
22          BGPHeader.length = 21;
23      }
24
25      public BGPNotificationPacket(DataInputStream in) throws IOException {
26          header = new BGPHeader(in);
27          if (header.type != BGPHeader.TYPE_NOTIFICATION) {
28            throw new IOException("Expected packet type NOTIFICATION, received type "+header.
                  type);
29          }
30          int bodylength = header.packetlength - BGPHeader.length;
31          byte packetbuffer[] = new byte[bodylength];
32          in.readFully(packetbuffer, 0, bodylength);
33          fromBytes(packetbuffer, 0);
34      }
35
36      public BGPNotificationPacket(BGPHeader h, DataInputStream in) throws IOException {
37          header = h;
38          int bodylength = header.packetlength - BGPHeader.length;
39          byte packetbuffer[] = new byte[bodylength];
40          in.readFully(packetbuffer, 0, bodylength);
41          fromBytes(packetbuffer, 0);
42      }
43
44      @Override
45      public String toString() {
46          String Errors[] = { "",
47              "Message Header Error",
48              "OPEN Message Error",
49              "UPDATE Message Error",
50              "Hold Timer Expired",
51              "Finite State Machine Error",
52              "Cease"
53          };
54
55          String MessageHeaderErrors[] = {"",
56              "Connection Not Synchronized.",
57              "Bad Message Length.",
58              "Bad Message Type."
59          };
60
61          String OPENMessageErrors[] = {"",
62              "Unsupported Version Number.",
63              "Bad Peer AS.",
64              "Bad BGP Identifier.",
65              "Unsupported Optional Parameter",
66              "Authentication Failure.",
67              "Unacceptable Hold Time."
68          };
69
70          String UPDATEMessageErrors[] = { "",
71              "Malformed Attribute List",
72              "Unrecognized Well-known Attribute",
73              "Missing Well-known Attribute",
74              "Attribute Flags Error",
75              "Attribute Length Error",
76              "Invalid ORIGIN Attribute",
77              "AS Routing Loop",
78              "Invalid NEXT_HOP Attribute",
79              "Optional Attribute Error",
80              "Invalid Network Field",
81              "Malformed AS_PATH"
82          };
83
84          String toreturn = "NOTIFICATION: "+Integer.toString(error)+"."+
85          Integer.toString(subcode)+": ";
86
87          if (error == MESSAGE_HEADER_ERROR) {
88              toreturn = toreturn + Errors[error];
89              if (subcode > 0 && subcode < MessageHeaderErrors.length) {
90                  toreturn = toreturn +": "+MessageHeaderErrors[subcode];
```

```
91              }
92          } else if (error == OPEN_MESSAGE_ERROR) {
93              toreturn = toreturn + Errors[error];
94              if (subcode > 0 && subcode < OPENMessageErrors.length) {
95                  toreturn = toreturn +": "+OPENMessageErrors[subcode];
96              }
97          } else if (error == UPDATE_MESSAGE_ERROR) {
98              toreturn = toreturn + Errors[error];
99              if (subcode > 0 && subcode < UPDATEMessageErrors.length) {
100                 toreturn = toreturn + ": " + UPDATEMessageErrors[subcode];
101             }
102         } else if (error == HOLD_TIMER_EXPIRED) {
103             toreturn = toreturn + Errors[error];
104         } else if (error == FINITE_STATE_MACHINE_ERROR) {
105             toreturn = toreturn + Errors[error];
106         } else if (error == CEASE) {
107             toreturn = toreturn + Errors[error];
108         } else {
109             toreturn = toreturn + "Unknown";
110         }
111         /* need to parse the data field into the error message eventually */
112         return toreturn;
113     }
114
115     public int length() {
116         int toreturn = 2; /* static data length */
117         if (data != null) {
118             toreturn = toreturn + data.length;
119         }
120         return toreturn;
121     }
122
123
124     public void manda(OutputStream out) throws IOException {
125         int l = length();
126         //System.out.println("Agregando "+l+" a "+BGPHeader.length);
127         header.packetlength = BGPHeader.length + l-2;
128         header.type=BGPHeader.TYPE_NOTIFICATION;
129
130         //header.toBytes(outbuffer, 19);
131         //System.out.println("Escribiendo encabezado "+header.toString());
132         header.writeTox(out);
133         //System.out.println("Envió No. 1 "+toString());
134
135         //out.write(outbuffer);
136     }
137
138     public void writeTo(OutputStream out) throws IOException {
139         int l = length();
140         header.packetlength = BGPHeader.length + l;
141         header.writeTo(out);
142         byte outbuffer[] = new byte[l];
143         toBytes(outbuffer, 0);
144         System.out.println("Envió No. 2 "+toString());
145         out.write(outbuffer);
146     }
147
148     public final void fromBytes(byte b[], int offset) {
149         int datalen = header.packetlength - 21;
150         data = new byte[datalen];
151         error = b[offset];
152         subcode= b[offset+1];
153         System.arraycopy(b, offset+2, data, 0, datalen);
154     }
155
156     public void toBytes(byte b[], int offset) {
157         System.out.println("Poniendo error "+error+" sub "+subcode+" en offset "+offset);
158         b[offset] = (byte) error;
159         b[offset+1] = (byte) subcode;
160         if (data != null) {
161             System.arraycopy(data, 0, b, offset+2, data.length);
```

```
162             }
163         }
164
165
166     }
```

Listado A.4: BGPOpenPacket.java

```
1   package bgp;
2   /* BGP OPEN packet */
3
4   import java.io.*;
5   import java.util.*;
6
7   class BGPOpenPacket {
8
9       public BGPHeader header;
10      public int version;
11      public int AS;
12      //public int ASe;
13      public int holdtime;
14      public long BGPID=0;
15      //public long BGPID_Interno=16909061;
16      public long BGPID_Interno=Long.parseLong("01020304", 16);  // ip=1.2.3.4
17      //http://www.cisco.com/c/en/us/support/docs/ip/border-gateway-protocol-bgp/13753-25.html
18      // por eso se define 1.2.3.4 para el paso 11  :)
19      public ArrayList params;
20
21      public static boolean debug = false;
22
23      public BGPOpenPacket() {
24          params = new ArrayList(0);
25          header = new BGPHeader();
26          header.type = BGPHeader.TYPE_OPEN;
27          header.packetlength = BGPHeader.length + length();
28      }
29
30      BGPOpenPacket(DataInputStream in) throws IOException {
31          if (debug) System.out.println("OPEN: Waiting for header");
32              header = new BGPHeader(in);
33          if (header.type != BGPHeader.TYPE_OPEN) {
34              throw new IOException("Expected packet type OPEN, received type "+header.type);
35          }
36          int bodylength = header.packetlength - BGPHeader.length;
37          byte packetbuffer[] = new byte[bodylength];
38          if (debug) System.out.println("OPEN: Waiting for body");
39          in.readFully(packetbuffer, 0, bodylength);
40          if (debug) System.out.println("OPEN: received body");
41          fromBytes(packetbuffer, 0);
42      }
43
44
45      BGPOpenPacket(BGPHeader h, DataInputStream in) throws IOException {
46          header = h;
47          int bodylength = header.packetlength - BGPHeader.length;
48          byte packetbuffer[] = new byte[bodylength];
49          in.readFully(packetbuffer, 0, bodylength);
50          fromBytes(packetbuffer, 0);
51      }
52
53      public void toBytes(byte b[], int offset) {
54          b[offset] = (byte)version;
55
56          //System.out.println("AS_r "+ASe);
57          //System.out.println("AS "+AS);
58
59
60          b[offset+1] = util.byteOfInt(AS, 1);
61          b[offset+2] = util.byteOfInt(AS, 0);
```

```
62          b[offset+3] = util.byteOfInt(holdtime, 1);
63          b[offset+4] = util.byteOfInt(holdtime, 0);
64          //System.out.println("+-+-*+-"+BGPID_Interno+"*+*+*+*+*+");
65          b[offset+5] = util.byteOfLong(BGPID_Interno, 3);
66          b[offset+6] = util.byteOfLong(BGPID_Interno, 2);
67          b[offset+7] = util.byteOfLong(BGPID_Interno, 1);
68          b[offset+8] = util.byteOfLong(BGPID_Interno, 0);
69          int pcount = 0;
70          int poffset = offset+10;
71          //for (Enumeration p = params.elements(); p.hasMoreElements(); ) {
72          for (int indi=0;indi<params.size();indi++) {
73              BGPOpenParameter param = (BGPOpenParameter)params.get(indi);
74              if (debug) System.out.println("Mandando "+util.convertToHexString(param.value));
75              param.toBytes(b, poffset);
76              poffset = poffset+param.length();
77              pcount++;
78          }
79          b[offset+9] = (byte) (poffset-29);
80      }
81
82      public void writeTo(OutputStream out) throws IOException {
83          /*
84          if (false) {
85          // write out the header
86          int l = length();
87          header.packetlength = header.length + l;
88          header.writeTo(out);
89
90          // write out this packet
91          byte outbuffer[] = new byte[l];
92          toBytes(outbuffer, 0);
93          if (debug) System.out.println("Sending "+toString());
94          out.write(outbuffer);
95          out.flush();
96          if (debug) System.out.println("Sent "+toString());
97          } else {
98          */
99
100         int l = length();
101         header.packetlength = BGPHeader.length + l;
102         byte outbuffer[] = new byte[header.packetlength];
103         /* write out the header */
104         header.toBytes(outbuffer,0);
105
106         /* write out this packet */
107         //System.err.println("----->"+length()+" ("+BGPHeader.length+")");
108         toBytes(outbuffer, BGPHeader.length);
109         if (debug) System.out.println("Sending. "+toString());
110         //System.out.println("Sending "+toString());
111         out.write(outbuffer);
112         out.flush();
113         if (debug) System.out.println("Sent. "+toString());
114         //System.out.println("Sent "+toString());
115         //}
116     }
117
118     @Override
119     public String toString() {
120         String toreturn = "OPEN: ";
121         toreturn = toreturn + "version: "+Integer.toString(version);
122         toreturn = toreturn + " AS: "+Integer.toString(AS);
123         toreturn = toreturn + " holdtime: "+Integer.toString(holdtime);
124         toreturn = toreturn + " *BGPID: "+Long.toString(BGPID_Interno);
125         toreturn = toreturn + " (" + InetNetwork.toString(BGPID_Interno) + ")";
126         return toreturn;
127     }
128
129     final int length() {
130         int l = 10 ;    /* OPEN static data */
131         /* length of all the params */
```

```
132         //for(Enumeration p = params.elements(); p.hasMoreElements(); ) {
133         for (int indi=0;indi<params.size();indi++) {
134             l = l + ((BGPOpenParameter)(params.get(indi))).length();
135         }
136         return l;
137     }
138
139     public final void fromBytes(byte b[], int offset ) throws IOException {
140         /* get the packet */
141         version = b[offset];
142         System.out.println("Version "+version);
143         AS = util.intFromBytes((byte) 0,(byte) 0, b[offset+1], b[offset+2]);
144         //System.out.println("AS_r "+ASe);
145         //System.out.println("AS "+AS);
146         holdtime = util.intFromBytes((byte) 0, (byte) 0, b[offset+3], b[offset+4]);
147         BGPID = util.intFromBytes(b[offset+5], b[offset+6], b[offset+7], b[offset+8]);
148         System.out.println("BGP ID "+InetNetwork.toString(BGPID) );
149         int paramcount = b[offset+9];
150         //System.out.println("Num. Parámetros "+paramcount);
151         //paramcount=5;
152         params = new ArrayList();
153         int poffset = offset + 10;
154         // Version Inicial, con error por confundir Longitud con # de parametros
155         // Se cambia For por While...
156         /*
157         for (int i = 1; i < paramcount; i++) {
158           BGPOpenParameter param = new BGPOpenParameter();
159           System.out.println(poffset);
160           param.fromBytes(b, poffset);
161           params.addElement(param);
162           poffset = poffset + param.value.length + 2;
163         }
164         */
165         while (poffset < paramcount+10) {
166             BGPOpenParameter param = new BGPOpenParameter();
167             //if (debug) System.out.println(poffset+"<"+paramcount);
168             param.fromBytes(b, poffset);
169             params.add(param);
170             poffset = poffset + param.value.length + 2;
171         }
172         System.out.println("Num. Parámetros "+params.size());
173         if (debug) {
174             for (int i=0;i<params.size();i++) {
175                 BGPOpenParameter bop = (BGPOpenParameter) params.get(i);
176                 System.out.println((i+1)+".- "+util.convertToHexString(bop.value)+" "+bop.
                        DescCodigo()+" ("+bop.Codigo()+")");
177             }
178         }
179     }
180 }
```

Listado A.5: BGPOpenParameter.java

```
1  package bgp;
2
3
4  class BGPOpenParameter {
5
6      public int type;
7      public static int TYPE_AUTH = 1;
8      public byte[] value;
9
10     public int length() {
11         return 2 + value.length;
12     }
13
14     public void toBytes(byte b[], int offset) {
15         b[offset] = (byte) type;
```

```
16          b[offset +1] = (byte) value.length;
17          System.arraycopy(value, 0, b, offset+2, value.length);
18      }
19
20      public void fromBytes(byte b[], int offset) {
21          if (offset<b.length) {
22              type = b[offset];
23              //System.out.println("\t tipo "+type);
24              int valuelen = b[offset+1];
25              //System.out.println("\t Valor Len"+valuelen);
26              value = new byte[valuelen];
27              System.arraycopy(b, offset+2, value, 0, valuelen);
28          } else {
29              System.err.println("Offset Mayor"+offset+">"+b.length);
30          }
31      }
32
33      public int Codigo() {
34          //https://www.iana.org/assignments/capability-codes/capability-codes.xhtml
35          return value[0] & 0xFF;
36      }
37
38      public String afi(byte[] value) {
39          int AFI=util.intFromBytes(value[2], value[2], value[2], value[3]);
40          int SAFI=value[5] & 0xFF;
41
42          return "AFI="+AFI+" SAFI="+SAFI;
43
44      }
45
46      public String DescCodigo() {
47
48          //https://www.iana.org/assignments/bgp-parameters/bgp-parameters.xhtml
49
50          String Salida;
51
52          int numero=(value[0] & 0xFF);
53
54          switch(numero){
55          case 0: Salida="Reservado";
56                  break;
57          case 1: Salida="Multiprotocol Extensions for BGP-4 "+afi(value);
58                  break;
59          case 2: Salida="Route Refresh Capability for BGP-4";
60                  break;
61          case 3: Salida="Outbound Route Filtering Capability";
62                  break;
63          case 4: Salida="Multiple routes to a destination capability (deprecated)";
64                  break;
65          case 5: Salida="Extended Next Hop Encoding";
66                  break;
67          case 6: Salida="BGP-Extended Message (TEMPORARY - registered 2015-09-30, expires 2018-
                09-30)";
68                  break;
69          case 7: Salida="BGPsec Capability";
70                  break;
71          case 8: Salida="Multiple Labels Capability";
72                  break;
73          case 64: Salida="Graceful Restart Capability";
74                  break;
75          case 65: Salida="Support for 4-octet AS number capability AS"+util.
                intFromBytes(value[2], value[3], value[4], value[5]);
76                  break;
77          case 67: Salida="Support for Dynamic Capability (capability specific)";
78                  break;
79          case 68: Salida="Multisession BGP Capability";
80                  break;
81          case 69: Salida="ADD-PATH Capability";
82                  break;
83          case 70: Salida="Enhanced Route Refresh Capability";
84                  break;
85          case 71: Salida="Long-Lived Graceful Restart (LLGR) Capability";
86                  break;
```

```
87          case 72: Salida="Unassigned";
88                  break;
89          case 73: Salida="FQDN Capability";
90                  break;
91          case 128: Salida="Reservado Cisco-Router Refresh 128";
92                  break;
93          case 131: Salida="Reservado Cisco-Multisession";
94                  break;
95          case 132: Salida="Reservado Cisco ";
96                  break;
97
98          default: Salida="Parámetro no encontrado ("+numero+")";
99          //https://www.iana.org/assignments/capability-codes/capability-codes.xhtml
100         }
101         return Salida;
102     }
103
104 }
```

Listado A.6: BGPPathAttribute.java

```
1  package bgp;
2
3
4  class BGPPathAttribute {
5
6      public boolean optional;
7      public boolean transitive;
8      public boolean partial;
9
10     /* Note that while we correctly identify the known types, we don't make
11     handling them particularly easy. TYPE_AS_PATH, for instance, implies
12     a Vector of AS path segments */
13
14     public int type;
15     public static int TYPE_ORIGIN = 1;
16     public static int TYPE_AS_PATH = 2;
17     public static int TYPE_NEXT_HOP = 3;
18     public static int TYPE_MULTI_EXIT_DISC = 4;
19     public static int TYPE_LOCAL_PREF = 5;
20     public static int TYPE_ATOMIC_AGGREGATE = 6;
21     public static int TYPE_AGGREGATORT = 7;
22
23     public static int TYPE_Multiprotocol_Reachable_NLRI=14;
24     public static int TYPE_Multiprotocol_Unreachable_NLRI=15;
25     public static int TYPE_AS4_PATH = 17;
26     public static int TYPE_AS4_AGGREGATORT = 18;
27     public int length; // data.length + flag byte + type byte + length byte(s)
28     public byte data[];
29
30     public void fromBytes(byte b[], int offset) {
31         optional =   ((b[offset] & 0x80) != 0);
32         transitive = ((b[offset] & 0x40) != 0);
33         partial =    ((b[offset] & 0x20) != 0);
34         type = b[offset+1];
35         if ((b[offset] & 0x10) != 0) { // extended length bit set
36             length = util.intFromBytes((byte)0, (byte)0, b[offset+2], b[offset+3]);
37             data = new byte[length];
38             System.arraycopy(b, offset+4, data, 0, length);
39             length = length + 4;
40         } else {                        // extended length bit NOT set
41             length = util.intFromBytes((byte)0, (byte)0, (byte)0, b[offset+2]);
42             data = new byte[length];
43             System.arraycopy(b, offset+3, data, 0, length);
44             length = length + 3;
45         }
46     }
47
48     public void toBytes (byte b[], int offset) {
49         /*
```

```
50          byte flags = 0;
51          if (optional) {
52            flags = (byte) (flags | 0x80);
53          }
54          if (transitive) {
55            flags = (byte) (flags | 0x40);
56          }
57          if (partial) {
58            flags = (byte) (flags | 0x20);
59          }
60          if (length > 255) {
61            flags = (byte) (flags | 0x10);
62          }
63          */
64          b[offset+1] = (byte) type;
65          if (length > 255) {
66              b[offset+2] = util.byteOfInt(length, 1);
67              b[offset+3] = util.byteOfInt(length, 0);
68              System.arraycopy(data, 0, b, offset+4, length);
69          } else {
70              b[offset+2] = (byte) length;
71              System.arraycopy(data, 0, b, offset+3, length);
72          }
73      }
74  }
```

Listado A.7: BGPSendUpdatePacket.java

```
1   package bgp;
2   /* BGP OPEN packet */
3   // 8 Enero 2018 error en bytes de nlri ipv4
4
5   import static bgp.util.LengNumto1String;
6   import static bgp.util.LengNumto4String;
7   import java.io.*;
8   import java.net.InetAddress;
9   import java.net.UnknownHostException;
10  import java.util.*;
11
12  public class BGPSendUpdatePacket {
13
14      BGPHeader header;
15      public int version;
16      public int AS;
17      public int holdtime;
18      public long BGPID;
19      public ArrayList params;
20
21      public static boolean debug = false;
22
23      private String W="";
24      public String W_red="";
25      public int W_prefijo=0;
26
27
28      private final String Path_Origen;
29      public String Path_ASPathList    ="";
30      private final String Path_NextHop;
31      private final String Path_LocalPref_;
32      public int Path_LocalPref;
33      public InetAddress Path_NextHopIP;
34      private String Path_AS;
35
36
37      public int NLRI_PrefijoRed=0;
38      public String NLRI_RedIP="";
39
40      public BGPSendUpdatePacket() {
41          params = new ArrayList(0);
42          header = new BGPHeader();
```

```
43          header.type = BGPHeader.TYPE_UPDATE;
44          //header.packetlength = BGPHeader.length + length();
45          Path_Origen    ="40010100";
46          Path_LocalPref_="40050400000000";
47          Path_NextHop="40030400000000";
48
49      }
50
51      public int toBytesW(byte b[], int offset) {
52          String Len="0000";
53
54          if ((W_red.length()>0) && (W_prefijo!=0)) {
55
56              String[] redIP =W_red.split("\\.");
57              int[] octeto=new int[redIP.length];
58              for (int octetos=0;octetos<redIP.length;octetos++) {
59                  octeto[octetos]=Integer.parseInt(redIP[octetos]);
60                  //System.out.println(redIP[octetos]);
61              }
62
63              String Red="";
64              for (int octetos=0;octetos<3;octetos++) {
65                  Red=Red+util.LengNumto1String(octeto[octetos]);
66              }
67
68              String Prefijo=util.LengNumto1String(W_prefijo);
69              W=Prefijo+Red;
70          } else {
71              W="";
72          }
73
74          byte[] p=util.toByteArray(Len+W);
75
76          p[0]=util.byteOfInt(p.length-2, 1);
77          p[1]=util.byteOfInt(p.length-2, 0);
78          System.arraycopy(p,0,b,offset,p.length);
79
80          return offset+p.length;
81      }
82
83      public int toBytesPath(byte b[], int offset) {
84
85          String Len       ="0000";
86          byte[] LocalPref=util.toByteArray(Path_LocalPref_);
87          LocalPref[3]=util.byteOfInt(Path_LocalPref, 3);
88          LocalPref[4]=util.byteOfInt(Path_LocalPref, 2);
89          LocalPref[5]=util.byteOfInt(Path_LocalPref, 1);
90          LocalPref[6]=util.byteOfInt(Path_LocalPref, 0);
91          byte[] nexthop=util.toByteArray(Path_NextHop);
92          nexthop[3]=Path_NextHopIP.getAddress()[0];
93          nexthop[4]=Path_NextHopIP.getAddress()[1];
94          nexthop[5]=Path_NextHopIP.getAddress()[2];
95          nexthop[6]=Path_NextHopIP.getAddress()[3];
96          Path_AS=ProcesaPath(Path_ASPathList);
97          //System.out.println(Path_AS);
98
99          byte[] p=util.toByteArray(Len+Path_Origen+Path_AS+util.toHexString(nexthop)+util.
                toHexString(LocalPref));
100         p[0]=util.byteOfInt(p.length-2, 1);
101         p[1]=util.byteOfInt(p.length-2, 0);
102         System.arraycopy(p,0,b,offset,p.length);
103
104         return offset+p.length;
105     }
106
107     public int toBytesNLRI(byte b[], int offset) {
108
109         byte[] p;
110
111         //System.out.println("xzxzxzxzx "+NLRI_RedIP+" "+NLRI_PrefijoRed);
112         if (NLRI_RedIP.contains(".")) {
113             //Para IPv4
```

```
114             String[] redIP =NLRI_RedIP.split("\\.");
115             int[] octeto=new int[redIP.length];
116 
117             for (int octetos=0;octetos<redIP.length;octetos++) {
118                 octeto[octetos]=Integer.parseInt(redIP[octetos]);
119                 //System.out.println(redIP[octetos]);
120             }
121             // Cambio enero 2018 para enviar solo los bytes requeridos en en NLRI
122             String Red="";
123             for (int octetos=0;octetos<(int) (0.9+NLRI_PrefijoRed/8.0);octetos++) {
124                 Red=Red+util.LengNumto1String(octeto[octetos]);
125             }
126 
127             String Prefijo=util.LengNumto1String(NLRI_PrefijoRed);
128 
129             p=util.toByteArray(Prefijo+Red);
130             System.arraycopy(p,0,b,offset,p.length);
131         } else {
132             //Para IPv6
133             String[] redIP =NLRI_RedIP.split("\\:");
134             int[] octeto=new int[6];
135             //System.out.println("cvcvcvc"+redIP.length*4);
136 
137             //for (int octetos=0;octetos<redIP.length;octetos++) {
138             //    octeto[octetos]=Integer.parseInt(redIP[octetos]);
139                 //System.out.println(redIP[octetos]);
140             //}
141             octeto[0]=28;
142             octeto[1]=1;
143             octeto[2]=0;
144             String hex = "ff";
145             octeto[3]= Integer.parseInt(hex, 16);
146             octeto[4]= Integer.parseInt("ba", 16);
147             octeto[5]= Integer.parseInt("ab", 16);
148 
149 
150             String Prefijo=util.LengNumto1String(NLRI_PrefijoRed);
151             String Red="280100ff";
152 
153             p=util.toByteArray(Prefijo+Red);
154             System.arraycopy(p,0,b,offset,p.length);
155 
156         }
157         return offset+p.length;
158     }
159 
160 
161     public void writeTo(OutputStream out) throws IOException {
162 
163 
164         int l = length();
165         header.packetlength = BGPHeader.length + l;
166 
167         byte outbuffer[] = new byte[header.packetlength];
168         /* write out the header */
169         header.toBytes(outbuffer,0);
170         // Escribimo W
171         l=toBytesW(outbuffer,BGPHeader.length);
172         // Escribimos Path
173         if (W.length()==0) {
174             l=toBytesPath(outbuffer, l);
175 
176             // Escribimos NLRI
177             toBytesNLRI(outbuffer, l);
178         }
179 
180         if (debug) System.out.println("Sending. "+toString());
181         //System.out.println("Sending "+toString());
182         out.write(outbuffer);
183         out.flush();
184         if (debug) System.out.println("Sent. "+toString());
```

```
185             //System.out.println("Sent "+toString());
186         //}
187         }
188
189         public void writeTo6(OutputStream out) throws IOException {
190
191
192             int l = length();
193             //System.out.println("Largo="+l);
194             header.packetlength = BGPHeader.length + l;
195
196             byte outbuffer[] = new byte[header.packetlength];
197             /* write out the header */
198             header.toBytes(outbuffer,0);
199
200             String WRL="0000"+util.LengNumto2String(l-4);
201             //String Fijo="800e"+"1a"+"000201";
202             String NH="10"+util.IPv6(Path_NextHopIP)+"00";    //  Len=18 o 36 caracteres
203             String NLRI=LengNumto1String(NLRI_PrefijoRed)+util.Segmento(InetAddress.getByName(NLRI_
                    RedIP), NLRI_PrefijoRed);
204
205             String Fijo="800e"+util.LengNumto1String(((NH.length()+NLRI.length())/2+3))+"000201";
206             String Origen="40010100";
207             String AS_Path=ProcesaPath(Path_ASPathList);
208             String MED="80040400000000";
209             String LP="400504"+LengNumto4String(Path_LocalPref);
210             String MP=WRL+Fijo+NH+NLRI+Origen+AS_Path+MED+LP;
211
212             //System.out.println(MP.length()/2);
213             //System.exit(0);
214
215             byte[] p=util.toByteArray(MP);
216             System.arraycopy(p,0,outbuffer,BGPHeader.length,p.length);
217
218
219             if (debug) System.out.println("Sending. "+toString());
220             //System.out.println("Sending "+toString());
221             out.write(outbuffer);
222             out.flush();
223             if (debug) System.out.println("Sent. "+toString());
224             //System.out.println("Sent "+toString());
225         //}
226         }
227
228
229
230         @Override
231         public String toString() {
232             String toreturn = "Send: ";
233             return toreturn;
234         }
235
236         final int length() {
237             int Salida=0;
238
239             if (util.esIpv6(Path_NextHopIP.toString())) {
240                 try {
241                     // Calculamos tamaño para Anuncio de IPv4
242                     String WRL="00000032";
243                     String NH="10"+util.IPv6(Path_NextHopIP)+"00";    //  Len=18 o 36 caracteres
244                     String NLRI=LengNumto1String(NLRI_PrefijoRed)+util.Segmento(InetAddress.
                            getByName(NLRI_RedIP), NLRI_PrefijoRed);
245                     String Fijo="800e"+util.LengNumto1String(((NH.length()+NLRI.
                            length())/2+3))+"000201";
246
247                     String Origen="40010100";
248                     String AS_Path=ProcesaPath(Path_ASPathList);
249                     String MED="80040400000000";
250                     String LP="400504"+"00000064";
251                     //System.out.println(NH);
252                     String MP=WRL+Fijo+NH+NLRI+Origen+AS_Path+MED+LP;
253
```

```
254                 Salida=MP.length()/2;
255             } catch (UnknownHostException ex) {
256                 System.err.println("leng err, en"+ex.toString());
257             }
258         } else {
259             // Calculamos tamaño para Anuncio de IPv4
260             int w=(W.length()/2);
261             //int NLRI=(NLRI_Prefijo.length()+NLRI_Red.length())/2;
262             int NLRI=4;
263             Path_AS=ProcesaPath(Path_ASPathList);
264             int Path_att=(Path_Origen.length()+Path_AS.length()+Path_NextHop.length()+Path_
                    LocalPref_.length())/2;
265 
266             int l = 2+w+2+Path_att+NLRI ;    /* UPDATE static data */
267             Salida= 2+w+2+Path_att+(int) (0.9+NLRI_PrefijoRed/8.0) +1;    /* UPDATE static data
                    ↪ */
268         }
269 
270         //System.out.println("final="+l +" ++ "+l2);
271         return Salida;
272     }
273 
274     private String ProcesaPath_ant(String p) {
275 
276         if (p.length()>0) {
277 
278         String[] Num_AS = p.split(" ");
279         //System.out.println(Num_AS.length);
280         int[] SA=new int[Num_AS.length];
281 
282         String segmentos="";
283         for (int num=0;num<Num_AS.length;num++) {
284             SA[num]=Integer.parseInt(Num_AS[num]);
285             //System.out.println("Camino "+(num+1)+".-"+SA[num]);
286             segmentos=segmentos+ util.ASnumtoString(SA[num]);
287         }
288 
289         String len=util.LengNumto2String(Num_AS.length*2+2);
290         String num=util.LengNumto1String(Num_AS.length);
291 
292         return "5002"+len+"02"+num+segmentos;
293         } else {
294             return "400200";
295         }
296     }
297 
298     private  String ProcesaPath(String p) {
299 
300         if (p.length()>0) {
301 
302         String[] Num_AS = p.split(" ");
303         //System.out.println(Num_AS.length);
304         int[] SA=new int[Num_AS.length];
305 
306 
307         String segmentos="";
308         for (int num=0;num<Num_AS.length;num++) {
309             SA[num]=Integer.parseInt(Num_AS[num]);
310             //System.out.println("Camino "+(num+1)+".-"+SA[num]);
311             segmentos=segmentos+ util.ASnumtoString32(SA[num]);
312         }
313 
314         String len=util.LengNumto1String(Num_AS.length*4+2);
315         //System.out.println("len="+len);
316         String num=util.LengNumto1String(Num_AS.length);
317 
318         return "4002"+len+"02"+num+segmentos;
319         } else {
320             return "400200";
321         }
322     }
323 
```

```
324
325
326 }
```

Listado A.8: BGPStatus.java

```
1  package bgp;
2
3  import java.util.*;
4  import java.util.concurrent.ConcurrentHashMap;
5
6  public class BGPStatus  {
7
8      public long BGPID; /* init'd by the constructor */
9
10     Date enteredcurrentstate;
11     public long updates = 0;
12     public long withdrawls = 0;
13     public long announces = 0;
14     //public ArrayList<Ruta> routes = new ArrayList<>();
15     public Map<String,Ruta> routes = new ConcurrentHashMap<>();
16
17
18     public BGPStatus(long newBGPID) {
19         BGPID = newBGPID;
20         resetUptime();
21     }
22
23     public long BGPID() {
24         return BGPID;
25     }
26
27     public long uptime() {
28         return ((new Date()).getTime() - enteredcurrentstate.getTime());
29     }
30
31     public final void resetUptime() {
32         enteredcurrentstate = new Date();
33     }
34
35     public void addWithdrawl(InetNetwork withdrawl) {
36         withdrawls++;
37         routes.remove(withdrawl.toString());
38     }
39
40     public void addAnnounce(InetNetwork announce, String NextHop) {
41         announces++;
42         Ruta r= new Ruta(announce.toString(), NextHop);
43         routes.put(announce.toString(), r);
44     }
45
46
47     public void addAnnounce(InetNetwork announce, String NextHop, String Path) {
48         announces++;
49         Ruta r= new Ruta(announce.toString(), NextHop, Path);
50         routes.put(announce.toString(), r);
51     }
52
53 }
```

Listado A.9: BGPUpdatePacket.java

```
1  package bgp;
2
3  import java.util.*;
4  import java.io.*;
5
```

```
6  class BGPUpdatePacket {
7
8      private final boolean debug=false;
9
10     public BGPHeader header;
11
12     public Vector withdrawls;
13     public Vector pathattrs;
14     public Vector nlrinfo;
15
16     public String NextHop;
17     public String Paths="*";
18     public EstructuraIPv6 DIPv6=new EstructuraIPv6();
19
20     public BGPUpdatePacket(BGPHeader h, DataInputStream in) throws IOException {
21         header = h;
22         int bodylength = header.packetlength - header.length;
23         byte packetbuffer[] = new byte[bodylength];
24         in.readFully(packetbuffer, 0, bodylength);
25         NextHop= fromBytes(packetbuffer, 0);
26         //Path=Paths(packetbuffer, 0);
27     }
28
29     public String fromBytes(byte b[], int offset) {
30         if (debug) {
31             System.out.println("Recibiendo "+b.length+ " bytes... "+offset+" de offset");
32         }
33         String NH="";
34         withdrawls = new Vector();
35         nlrinfo = new Vector();
36         pathattrs = new Vector();
37
38         /* withdrawls */
39         int toffset = offset + 2;
40         int withdrawllen = util.intFromBytes((byte) 0, (byte) 0, b[offset], b[offset+1]);
41         if (debug) {
42             System.out.println("Withdrawlen = "+withdrawllen);
43         }
44
45         while (toffset < offset + withdrawllen) {
46             int prefixlen = (b[toffset] + 7) / 8;
47             byte prefix[] = new byte[4];
48             System.arraycopy(b, toffset+1, prefix, 0, prefixlen);
49             InetNetwork redb=new InetNetwork(prefix[0], prefix[1], prefix[2], prefix[3],
                   ↪ b[toffset]);
50             withdrawls.addElement(redb);
51             if (debug) {
52                 System.out.println("Borrando "+redb);
53             }
54             toffset = toffset + prefixlen + 1;
55         }
56
57
58         /* path attributes */
59         int pathattrlen = util.intFromBytes((byte) 0, (byte) 0, b[toffset], b[toffset+1]);
60         toffset = toffset + 2; /* past the pathattrlen */
61         int nlroffset = toffset + pathattrlen; /* where the NRL info is */
62         BGPPathAttribute pathattr = null;
63         while (toffset < nlroffset) {
64             pathattr = new BGPPathAttribute();
65
66             //System.out.println(util.toHexString(b)+"-"+toffset);
67
68             pathattr.fromBytes(b, toffset);
69             pathattrs.addElement(pathattr);
70             toffset = toffset + pathattr.length;
71
72             //System.out.println("Tipo att = "+pathattr.type);
73             //System.out.println("Tipo s = "+pathattr.toString());
74
75             if (pathattr.type==BGPPathAttribute.TYPE_NEXT_HOP) {
76                 long ip=util.intFromBytes(pathattr.data[0], pathattr.data[1], pathattr.data[2],
```

```
                    ↪ pathattr.data[3]);
77                  if (debug) {
78                      // System.out.println((pathattr.data[0]  & 0xfF)+ " " + (pathattr.data[1]&
                            ↪  0xfF)+ " " + (pathattr.data[2]&  0xfF)+ " " + (pathattr.data[3]&
                            ↪ 0xfF)+ " " );
79                  }
80                  //System.out.println("next hop "+InetNetwork.toString(ip)+" **"+ip+"**");
81                  NH=InetNetwork.toString(ip);
82              }
83
84              if (pathattr.type==BGPPathAttribute.TYPE_Multiprotocol_Reachable_NLRI) {
85                  DIPv6.Agrega();
86                  //long ip=util.intFromBytes(pathattr.data[0], pathattr.data[1], pathattr.
                        data[2], pathattr.data[3]);
87                  if (debug) {
88                      System.out.println((pathattr.data[0]  &  0xfF)+ " " + (pathattr.data[1]&
                            ↪ 0xfF)+ " " + (pathattr.data[2]&  0xfF)+ " " + (pathattr.data[3]&
                            ↪ 0xfF)+ " " );
89                  }
90                  //System.out.println("14:: "+util.convertToHexString(pathattr.data));
91                  //NH=InetNetwork.toString(ip);
92                  int AFI=util.intFromBytes(pathattr.data[0], pathattr.data[0], pathattr.data[0],
                        ↪ pathattr.data[1]);
93                  int SAFI=pathattr.data[2] & 0xFF;
94                  if (debug) System.out.println("AFI="+AFI+"  SAFI="+SAFI);
95                  if (AFI==2 && SAFI==1) {
96                      int cant=(pathattr.data[3] & 0xFF) /16;
97                      //byte dirIPv6[] = new byte[16];
98                      for (int i=0;i<cant;i++) {
99                          byte dirIPv6[] = new byte[16];
100                         System.arraycopy(pathattr.data, 4+(i*16), dirIPv6, 0, dirIPv6.length);
101                         if (debug) System.out.println("NextHp"+(i+1)+".- "+util.
                                convertToHexString(dirIPv6));
102
103                         if (i==0) DIPv6.NextHop.add(new InetNetwork(util.
                                convertToHexString(dirIPv6),128));
104                     }
105
106                     int resto=pathattr.data.length-cant*16-5;
107                     if (debug) System.out.println("Resto="+resto);
108                     byte[] NLRI = new byte[resto];
109                     System.arraycopy(pathattr.data, 5+(cant*16), NLRI, 0, NLRI.length);
110                     if (debug) System.out.println("NRL "+util.toHexString(NLRI));
111
112                     while (NLRI.length>0) {
113                         int prefijo=NLRI[0] & 0xFF;
114                         if (debug) System.out.println("prefijo "+prefijo);
115                         int lsegmento=(int) (0.9+prefijo/8.0);  // tomo el segmento de 8 mas
                                ↪ proximo superior
116                         //int lsegmento=(int) Math.round(prefijo/8.0);  // tomo el segmento de
                                ↪ 8 mas proximo superior
117                         if (debug) System.out.println("Largo ="+lsegmento);
118                         byte[] segmento = new byte[lsegmento];
119                         System.arraycopy(NLRI, 1, segmento, 0, segmento.length);
120                         if (debug) System.out.println(util.toHexString(segmento)+"/"+prefijo);
121                         NLRI=removeNLRI(NLRI, lsegmento+1);
122                         DIPv6.AgregaSegmento(new InetNetwork(util.toHexString(segmento),
                                ↪ prefijo));
123
124                     }
125
126
127                 }
128
129             }
130             if (pathattr.type==BGPPathAttribute.TYPE_Multiprotocol_Unreachable_NLRI) {
131                 DIPv6.Borra();
132                 //long ip=util.intFromBytes(pathattr.data[0], pathattr.data[1], pathattr.
                        data[2], pathattr.data[3]);
133                 if (debug) {
```

```
134                     //System.out.println((pathattr.data[0]  & 0xfF)+ " " + (pathattr.data[1]&
                            ↪ 0xfF)+ " " + (pathattr.data[2]&  0xfF)+ " " + (pathattr.data[3]&
                            ↪ 0xfF)+ " " );
135                 }
136                 //System.out.println("15:: "+util.convertToHexString(pathattr.data));
137                 //NH=InetNetwork.toString(ip);
138                 int AFI=util.intFromBytes(pathattr.data[0], pathattr.data[0], pathattr.data[0],
                        ↪ pathattr.data[1]);
139                 int SAFI=pathattr.data[2] & 0xFF;
140                 if (debug) System.out.println("AFI="+AFI+"  SAFI="+SAFI);
141                 if (AFI==2 && SAFI==1) {
142
143                     int resto=pathattr.data.length-3;  // quitamos AFI Y SAFI
144                     if (debug) System.out.println("Resto="+resto);
145                     byte[] NLRI = new byte[resto];
146                     System.arraycopy(pathattr.data, 3, NLRI, 0, NLRI.length);
147                     if (debug) System.out.println("NRL "+util.toHexString(NLRI));
148
149                     while (NLRI.length>0) {
150                         int prefijo=NLRI[0] & 0xFF;
151                         if (debug) System.out.println("prefijo "+prefijo);
152                         int lsegmento=(int) (0.9+prefijo/8.0);  // tomo el segmento de 8 mas
                                ↪ proximo superior
153                         //int lsegmento=(int) Math.round(prefijo/8.0);  // tomo el segmento de
                                ↪ 8 mas proximo superior
154                         if (debug) System.out.println("Largo ="+lsegmento);
155                         byte[] segmento = new byte[lsegmento];
156                         System.arraycopy(NLRI, 1, segmento, 0, segmento.length);
157                         if (debug) System.out.println(util.toHexString(segmento)+"/"+prefijo);
158                         NLRI=removeNLRI(NLRI, lsegmento+1);
159                         DIPv6.AgregaSegmento(new InetNetwork(util.toHexString(segmento),
                                ↪ prefijo));
160
161                     }
162
163
164                 }
165
166             }
167
168             if (pathattr.type==BGPPathAttribute.TYPE_AS_PATH) {
169
170                 Paths=util.segmentASs(pathattr.data);
171                 Paths=Paths.trim();
172             }
173
174             if (debug) {
175                 if (pathattr.type>0) {
176                     System.out.println("Tipo att = "+descTipo(pathattr.type)+"("+ pathattr.
                            type+")");
177                 }
178             }
179         }
180
181
182         if (toffset != nlroffset) {
183             System.err.println("WARNING: INCONSISTENCY IN PACKET DATA");
184             toffset = nlroffset;
185         }
186
187         /* Network Layer Reachability (NLR) info */
188
189         while( toffset < offset + header.packetlength - BGPHeader.length) {
190             int prefixlen = (b[toffset] + 7) / 8;
191             if (prefixlen > 4) {
192                 System.err.println("ERROR: INVALID PREFIX LENGTH ("+Byte.
                        toString(b[toffset])+")");
193                 util.dumpArray(b,0,b.length-1);
194                 System.exit(1);
195                 b[toffset] = 32;
196             }
```

```
197             byte prefix[] = new byte[4];
198             System.arraycopy(b, toffset+1, prefix, 0, prefixlen);
199             nlrinfo.addElement(new InetNetwork(prefix[0], prefix[1], prefix[2], prefix[3],
                    ↪ b[toffset]));
200             toffset = toffset + prefixlen + 1;
201         }
202         return NH;
203     }
204
205     public void toBytes(byte b[], int offset) {
206
207         /* write out withdrawls */
208         int toffset = offset+2;
209         for(Enumeration w = withdrawls.elements(); w.hasMoreElements();) {
210             InetNetwork net = (InetNetwork) w.nextElement();
211             b[toffset] = (byte) net.prefixlength;
212             for (int i = 0; i < (b[toffset] / 8); i++) {
213                 b[toffset+1+i] = util.byteOfLong(net.prefix, 3-i);
214             }
215             toffset = toffset + 1 + (b[toffset] / 8);
216         }
217         b[offset] = util.byteOfInt(toffset - offset - 2, 1);
218         b[offset+1] = util.byteOfInt(toffset - offset - 2, 0);
219
220         /* write out the path attributes */
221         int pathattroff = toffset;
222         for(Enumeration p = pathattrs.elements(); p.hasMoreElements();) {
223             BGPPathAttribute pathattr = (BGPPathAttribute) p.nextElement();
224             pathattr.toBytes(b, toffset);
225             toffset = toffset + pathattr.length;
226         }
227         b[pathattroff] = util.byteOfInt(toffset - pathattroff - 2, 1);
228         b[pathattroff+1] = util.byteOfInt(toffset - pathattroff - 2, 0);
229
230         /* write out the NLR info */
231         for(Enumeration n = nlrinfo.elements(); n.hasMoreElements();) {
232             InetNetwork net = (InetNetwork) n.nextElement();
233             b[toffset] = (byte) net.prefixlength;
234             for (int i = 0; i < (b[toffset] / 8); i++) {
235                 b[toffset+1+i] = util.byteOfLong(net.prefix, 3-i);
236             }
237             toffset = toffset + 1 + (b[toffset] / 8);
238         }
239     }
240
241     public void ProcesaAtributos() {
242         for (int elem=0;elem<pathattrs.size();elem++) {
243             BGPPathAttribute pa=(BGPPathAttribute) pathattrs.get(elem);
244             System.out.println((elem+1)+".- "+util.toHexString(pa.data)+"("+pa.type+")");
245         }
246     }
247
248     public String descTipo(int numero) {
249         String Salida;
250
251         switch(numero){
252         case 0: Salida="Error";
253                 break;
254         case 1: Salida="ORIGIN*";
255                 break;
256         case 2: Salida="AS_PATH*";
257                 break;
258         case 3: Salida="NEXT_HOP*";
259                 break;
260         case 4: Salida="MULTI_EXIT_DISC (MED)";
261                 break;
262         case 5: Salida="LOCAL_PREF";
263                 break;
264         case 6: Salida="ATOMIC_AGGREGATE";
265                 break;
266         case 7: Salida="AGGREGATOR";
267                 break;
```

```
268             case 8: Salida="COMMUNITY";
269                     break;
270             case 9: Salida="ORIGINATOR_ID";
271                     break;
272             case 10: Salida="Cluster List";
273                     break;
274             case 11: Salida="DPA";
275                     break;
276             case 12: Salida="Advertiser";
277                     break;
278             case 13: Salida="RCID_PATH/CLUSTER_ID";
279                     break;
280             case 14: Salida="Multiprotocol Reachable NLRI";
281                     break;
282             case 15: Salida="Multiprotocol Unreachable NLRI";
283                     break;
284             case 16: Salida="Extended communities";
285                     break;
286             case 17: Salida="AS4_PATH";
287                     break;
288             case 18: Salida="AS4_AGGREGATOR";
289                     break;
290             case 19: Salida="SAFI Specific Attribute (SSA) (deprecated)";
291                     break;
292             case 20: Salida="Connector Attribute (deprecated)";
293                     break;
294             case 21: Salida="AS_PATHLIMIT (deprecated)";
295                     break;
296             case 22: Salida="PMSI_TUNNEL";
297                     break;
298             case 23: Salida="Tunnel Encapsulation Attribute";
299                     break;
300             case 24: Salida="Traffic Engineering";
301                     break;
302             case 25: Salida="IPv6 Address Specific Extended Community";
303                     break;
304             case 26: Salida="AIGP";
305                     break;
306             case 27: Salida="PE Distinguisher Labels";
307                     break;
308             case 28: Salida="BGP Entropy Label Capability Attribute (deprecated)";
309                     break;
310             case 29: Salida="BGP-LS Attribute";
311                     break;
312             case 30: Salida="Deprecated";
313                     break;
314             case 31: Salida="Deprecated";
315                     break;
316             case 32: Salida="LARGE_COMMUNITY";
317                     break;
318             case 33: Salida="BGPsec_Path";
319                     break;
320             case 34: Salida="BGP Community Container Attribute";
321                     break;
322             case 40: Salida="BGP Prefix-SID";
323                     break;
324             case 128: Salida="ATTR_SET";
325                     break;
326
327             default: Salida="Parámetro no encontrado ("+numero+")";
328             //https://www.iana.org/assignments/bgp-parameters/bgp-parameters.xhtml#bgp-parameters-1
329             }
330             return Salida;
331         }
332
333         public static byte[] removeNLRI(byte[] original, int element){
334             byte[] n = new byte[original.length - element];
335             System.arraycopy(original, element, n, 0, n.length );
336             //System.arraycopy(original, element+1, n, element, original.length - element-1);
337         return n;
338     }
339
340     }
```

Listado A.10: InetNetwork.java

```
1  package bgp;
2
3  public class InetNetwork {
4
5      public int prefixlength;
6      public long prefix;
7      public String IPv6;
8
9      public InetNetwork(String network, int masklen) {
10         IPv6=Acomoda(network);
11         prefixlength = masklen;
12         /* clear the low bits - probably not necessary */
13         //     prefix = (prefix & 0x0FFFFFFFFFFFFFFFL) << (32-masklen);
14     }
15
16     public InetNetwork(long network, int masklen) {
17         prefix = network;
18         prefixlength = masklen;
19         IPv6="";
20         /* clear the low bits - probably not necessary */
21         //     prefix = (prefix & 0x0FFFFFFFFFFFFFFFL) << (32-masklen);
22     }
23
24     public InetNetwork(byte o1, byte o2, byte o3, byte o4, int masklen) {
25         IPv6="";
26         prefix = util.longFromBytes((byte)0, (byte)0, (byte)0, (byte)0, o1, o2, o3, o4);
27         prefixlength = masklen;
28     }
29
30
31     public static String toString(long network, int masklen) {
32         InetNetwork n = new InetNetwork(network, masklen);
33         return n.toString();
34     }
35
36     public static String toString(long network) {
37         return toString(network, 32);
38     }
39
40     @Override
41     public String toString() {
42         String Salida;
43         if (IPv6.length()>1) {
44             if (prefixlength<128) {
45                 Salida=IPv6 + "/"+prefixlength;
46             } else {
47                 Salida=IPv6;
48             }
49         } else {
50             Salida=Long.toString((prefix & 0x00000000FF000000L) >> (long) 24)+"."+
51             Long.toString((prefix & 0x0000000000FF0000L) >> (long) 16)+"."+
52             Long.toString((prefix & 0x000000000000FF00L) >> (long) 8) +"."+
53             Long.toString( prefix & 0x00000000000000FFL)+"/"+
54             Integer.toString(prefixlength);
55         }
56         return Salida;
57     }
58
59     private String Acomoda(String n) {
60         String Salida="";
61         for (int i=1;i<=n.length();i++) {
62             Salida=Salida+n.charAt(i-1);
63             if ((i %4)==0) {
64                 Salida=Salida+":";
65             }
66         }
67         if (Salida.length()==40) {
68             Salida=Salida.substring(0,Salida.length()-1);
69         } else if (Salida.length()<36) {
```

```
70              Salida=Salida+":";
71          }
72          return Salida;
73      }
74  }
```

A.2. Código Java que implementa diferentes Listados de Rutas

Listado A.11: Listados.java

```
1   /*
2    * To change this license header, choose License Headers in Project Properties.
3    * To change this template file, choose Tools | Templates
4    * and open the template in the editor.
5    */
6   package bgp;
7
8   import Graficas.Path;
9   import analisis.SistemasAutonomos;
10  import java.net.InetAddress;
11  import java.net.UnknownHostException;
12  import java.text.SimpleDateFormat;
13  import java.util.ArrayList;
14  import java.util.Date;
15  import java.util.HashMap;
16  import java.util.HashSet;
17  import java.util.Iterator;
18  import java.util.Map;
19  import java.util.Set;
20  import java.util.SortedSet;
21  import java.util.TreeSet;
22  import up.varios.ColoresAnsi;
23
24  /**
25   *
26   * @author lelguea
27   * Modulos de Listas
28   */
29  public class Listados {
30
31      static int globalc;
32      static boolean debug=false;
33
34      static String borra1, borra2;
35
36      public static void Muestra_Stat_Resumen(Map Lista) {
37          //for(Enumeration e = Lista; e.hasMoreElements();) {
38          for (Iterator<String> conec = Lista.keySet().iterator(); conec.hasNext();) {
39              String conexion=conec.next();
40
41              System.out.println("Mostrando "+conexion);
42              BGPStatus status = (BGPStatus) Lista.get(conexion);
43
44              System.out.print(InetNetwork.toString(status.BGPID));
45              //System.out.print(" up for "+Long.toString(status.uptime() / 1000L)+"s");
46              //System.out.print(" up for "+util.formatoNum(status.uptime() / 1000.0 /3600/24)+"
                    ↪ days");
47
48              System.out.println(" up for "+util.Actividad(status.uptime())+".");
49
50              System.out.print("  Updates: "+util.formatoNum2(status.updates));
51              System.out.print("  Announces: "+util.formatoNum2(status.announces));
52              System.out.print("  Witdrawls: "+util.formatoNum2(status.withdrawls));
53              System.out.print("  Routes: "+util.formatoNum2(status.routes.size()));
54              System.out.println("");
55          }
56      }
57
58
```

```
59
60
61      public static ArrayList<BGPSendUpdatePacket>  Arregla4(Map<String,Ruta> rs) {
62          ArrayList<BGPSendUpdatePacket> Salida=new ArrayList<>();
63
64
65          Set<String> inicios = new HashSet<>();
66
67          HashMap<String,Integer> Vecinos = new HashMap<>();
68
69          HashMap<String,String> NextHopCorrectos = new HashMap<>();
70
71          rs.keySet().stream().map((key) -> {
72              if (util.ISP_Directo(rs.get(key).Path)) {
73                  if (debug) {
74                      System.out.println("El sa "+rs.get(key).Path+" sale por "+rs.get(key).
                            nextHop);
75                  }
76                  NextHopCorrectos.put(rs.get(key).Path, rs.get(key).nextHop);
77              }
78              return key;
79          }).map((key) -> {
80              if(Vecinos.containsKey(rs.get(key).nextHop)){
81                  int n=Vecinos.get(rs.get(key).nextHop) + 1;
82                  Vecinos.put(rs.get(key).nextHop, n);
83              } else {
84                  Vecinos.put(rs.get(key).nextHop,1);
85                  //System.out.println(rs.get(key).nextHop+ "++++++----++++++---->"+rs.get(key).
                        Path);
86              }
87              return key;
88          }).filter((key) -> (!rs.get(key).Path.contains("*"))).map((key) -> new Path(rs.get(key).
                Path)).forEachOrdered((p) -> {
89              //System.out.println("->"+rs.get(key).Path+"<-");
90              inicios.add(""+p.getSA().get(0));
91              //System.out.println("inicio size "+inicios.size());
92          });
93
94
95          Map<String, Integer> VecinosO = util.OrdenaMapa(Vecinos);
96
97          globalc=1;
98          System.out.println(ColoresAnsi.RED+"Buscando Errores por ISP");
99          SortedSet<String> llaveOrden = new TreeSet<>(rs.keySet());
100         int respa=1;
101         inicios.stream().forEach((sa) -> {
102
103             if (sa != null ) {
104                 try {
105                     if (debug) {
106                         System.out.println("\n"+sa);
107                         System.out.println("\n"+sa+" ("+SistemasAutonomos.getName(sa)+")");
108                         //System.out.println("\n"+sa+" ("+sa+")");
109                     }
110                     String nexthop_Correcto;
111                     nexthop_Correcto=NextHopCorrectos.get(sa.trim());
112                     // LME ERROR EN IPV6
113                     System.out.println("nh-> "+nexthop_Correcto+" sa="+sa.trim());
114                     if (nexthop_Correcto.indexOf("/")>0) {
115                         nexthop_Correcto=nexthop_Correcto.substring(0,nexthop_Correcto.
                                indexOf("/"));
116                         //System.out.println("Se encontró salida de "+sa.trim()+" por "+nexthop_
                                Correcto);
117                     }
118
119                     //System.out.println(ColoresAnsi.RED+"Rutas no optimizadas:");
120                     int contador=1;
121                     for ( String key3 : llaveOrden ) {
122                         if (rs.get(key3)!=null) {
123                             //System.out.println("*****++++------ "+key3+" de "+sa+" ..... "+rs.
                                    get(key3).Path);
```

```
124                             borra1=key3;
125                             borra2=rs.get(key3).Path;
126
127                             if (util.VecinoEnRuta(rs.get(key3).Path, 1, Integer.parseInt(sa))) {
128
129                                 String indice = String.format("%18s", (contador
                                        ↪ ++)+"/"+(globalc++)+"/" +rs.size()+".-");
130                                 String segmento = String.format("%20s", key3);
131                                 String nextHop = String.format("%35s","("+ rs.get(key3).
                                        nextHop+"->"+nexthop_Correcto +")");
132                                 //System.out.println((contador ++)+"/"+rs.size()+".- "+
                                        ↪ padded+" ("+rs.get(key).nextHop +") "+rs.get(key).
                                        Path+"    \t"+SistemasAutonomos.getName("13679"));
133                                 if (debug) {
134                                     System.out.println(indice+" "+ segmento+"  "+nextHop +"
                                            ↪ "+rs.get(key3).Path);
135                                 }
136                                 BGPSendUpdatePacket bsup=new BGPSendUpdatePacket();
137
138                                 bsup.NLRI_PrefijoRed= Integer.parseInt(key3.substring(key3.
                                        indexOf("/")+1));
139                                 bsup.NLRI_RedIP=key3.substring(0,key3.indexOf("/"));
140                                 //bsup.Path_ASPathList =rs.get(key3).Path;  //Arreglo muy burdo
141                                 bsup.Path_ASPathList =sa;
142                                 bsup.Path_NextHopIP=InetAddress.getByName(nexthop_Correcto);
143                                 bsup.Path_LocalPref=300;
144
145                                 Salida.add(bsup);
146                             }
147                         }
148                     }
149                 } catch (NumberFormatException | UnknownHostException e) {
150                     System.err.println("b "+e.toString());
151                     System.out.println("*****+++++------ "+borra1+"...."+borra2);
152                     //e.printStackTrace();
153                     //System.exit(-1);
154                 }
155             }
156
157         });
158
159         return Salida;
160     }
161
162     public static ArrayList<BGPSendUpdatePacket> Arregla6(Map<String,Ruta> rs) {
163
164         //Procesando 2001:0450::/32 que salga por 2001:450:2002:236::9D
165         // con path 3356 3549
166
167
168         ArrayList<BGPSendUpdatePacket> Salida=new ArrayList<>();
169         try {
170
171             BGPSendUpdatePacket bsup=new BGPSendUpdatePacket();
172             bsup.NLRI_PrefijoRed= 32;
173             bsup.NLRI_RedIP="2001:0450::";
174             //bsup.Path_ASPathList =rs.get(key3).Path;  //Arreglo muy burdo
175             bsup.Path_ASPathList ="";
176             bsup.Path_NextHopIP=InetAddress.getByName("2001:0450:2002:0236::9D");
177             bsup.Path_LocalPref=300;
178             //Salida.add(bsup);
179
180
181
182             BGPSendUpdatePacket bsup1=new BGPSendUpdatePacket();
183             bsup1.NLRI_PrefijoRed= 32;
184             bsup1.NLRI_RedIP="2002:0450::";
185             //bsup.Path_ASPathList =rs.get(key3).Path;  //Arreglo muy burdo
186             bsup1.Path_ASPathList ="";
187             bsup1.Path_NextHopIP=InetAddress.getByName("2001:0450:2002:0236::9D");
188             bsup1.Path_LocalPref=301;
189             //Salida.add(bsup1);
```

```
190
191             BGPSendUpdatePacket bsup2=new BGPSendUpdatePacket();
192             bsup2.NLRI_PrefijoRed= 48;
193             bsup2.NLRI_RedIP="2001:0450:2060::";
194             bsup2.Path_ASPathList ="3549";
195             bsup2.Path_NextHopIP=InetAddress.getByName("2001:0450:2002:0236::9D");
196             bsup2.Path_LocalPref=302;
197             Salida.add(bsup2);
198
199
200             return Salida;
201         } catch (UnknownHostException ex) {
202             System.err.println(ex.toString());
203         }
204
205         return Salida;
206     }
207
208     public static ArrayList<BGPSendUpdatePacket> Arregla6x(Map<String,Ruta> rs) {
209
210         //Procesando 2001:0450::/32 que salga por 2001:450:2002:236::9D
211         // con path 3356 3549
212
213
214         ArrayList<BGPSendUpdatePacket> Salida=new ArrayList<>();
215         try {
216
217             for (int i=32;i<65;i++) {
218                 BGPSendUpdatePacket bsup=new BGPSendUpdatePacket();
219                 bsup.NLRI_PrefijoRed= i;
220                 bsup.NLRI_RedIP="28"+i+":a0:30:c20:172:25:150:198";
221                 //bsup.Path_ASPathList =rs.get(key3).Path;  //Arreglo muy burdo
222                 bsup.Path_ASPathList ="13679 22434 "+i+i+i;
223                 bsup.Path_NextHopIP=InetAddress.getByName("2001:0450:2002:0236::9D");
224                 bsup.Path_LocalPref=300+i;
225                 Salida.add(bsup);
226             }
227
228
229             return Salida;
230         } catch (UnknownHostException ex) {
231             System.err.println(ex.toString());
232         }
233
234         return Salida;
235     }
236
237
238     public static ArrayList<BGPSendUpdatePacket> Arregla2(Map<String,Ruta> rs) {
239         ArrayList<BGPSendUpdatePacket> Salida=new ArrayList<>();
240
241         Set<String> inicios = new HashSet<>();
242
243         HashMap<String,Integer> Vecinos = new HashMap<>();
244
245         rs.keySet().stream().map((key) -> {
246             if(Vecinos.containsKey(rs.get(key).nextHop)){
247                 int n=Vecinos.get(rs.get(key).nextHop) + 1;
248                 Vecinos.put(rs.get(key).nextHop, n);
249             } else {
250                 Vecinos.put(rs.get(key).nextHop,1);
251             }
252             return key;
253         }).filter((key) -> (!rs.get(key).Path.contains("*"))).map((key) -> new Path(rs.get(key).
                Path)).forEachOrdered((p) -> {
254             //System.out.println("->"+rs.get(key).Path+"<-");
255             inicios.add(""+p.getSA().get(0));
256             //System.out.println("inicio size "+inicios.size());
257         });
258
259
260         //Comparator<String> comparator = new ComparadorValores<>(Vecinos);
```

```
261         //TreeMap<String, Integer> VecinosO = new TreeMap<>(comparator);
262         //VecinosO.putAll(Vecinos);
263         @SuppressWarnings("unchecked")
264         Map<String, Integer> VecinosO = util.OrdenaMapa(Vecinos);
265 
266         globalc=1;
267         System.out.println(ColoresAnsi.RED+"Buscando Errores por ISP");
268         SortedSet<String> llaveOrden = new TreeSet<>(rs.keySet());
269         int respa=1;
270         inicios.stream().forEach((sa) -> {
271 
272             if (sa != null ) {
273                 try {
274                     System.out.println("\n"+sa);
275                     System.out.println("\n"+sa+" ("+SistemasAutonomos.getName(sa)+")");
276                     //System.out.println("\n"+sa+" ("+sa+")");
277                     String nexthop_Correcto="";
278                     if (sa.contains("6503")) {
279                         nexthop_Correcto="148.245.154.1";
280                     }
281                     if (sa.contains("18734")) {
282                         nexthop_Correcto="189.202.194.198";
283                     }
284                     if (sa.contains("22566")) {
285                         nexthop_Correcto="201.161.34.97";
286                     }
287 
288                     System.out.println(ColoresAnsi.RED+"Rutas no optimizadas:");
289                     int contador=1;
290                     for ( String key3 : llaveOrden ) {
291 
292                         if (util.VecinoEnRuta(rs.get(key3).Path, 1, Integer.parseInt(sa))) {
293 
294                             String indice = String.format("%18s", (contador
                                    ↪ ++)+"/"+(globalc++)+"/" +rs.size()+".-");
295                             String segmento = String.format("%20s", key3);
296                             String nextHop = String.format("%35s","("+ rs.get(key3).nextHop+"-
                                    >"+nexthop_Correcto +")");
297                             //System.out.println((contador ++)+"/"+rs.size()+".- "+ padded+"
                                    ↪ ("+rs.get(key).nextHop +") "+rs.get(key).Path+"
                                    ↪ \t"+SistemasAutonomos.getName("13679"));
298                             System.out.println(indice+" "+ segmento+"  "+nextHop +" "+rs.
                                    get(key3).Path);
299                                         BGPSendUpdatePacket bsup=new
                                                ↪ BGPSendUpdatePacket();
300 
301                             bsup.NLRI_PrefijoRed= Integer.parseInt(key3.substring(key3.
                                    indexOf("/")+1));
302                             bsup.NLRI_RedIP=key3.substring(0,key3.indexOf("/"));
303                             bsup.Path_ASPathList =rs.get(key3).Path;
304                             bsup.Path_NextHopIP=InetAddress.getByName(nexthop_Correcto);
305                             bsup.Path_LocalPref=300;
306 
307                             Salida.add(bsup);
308 
309                         }
310                     }
311 
312                 } catch (NumberFormatException | UnknownHostException e) {
313                     System.err.println("b "+e.toString());
314                     //e.printStackTrace();
315                     //System.exit(-1);
316                 }
317             }
318         });
319 
320         return Salida;
321 
322     }
323 
324 
325 
```

```
326     public static void Detalle(Map<String,Ruta> rs) {
327
328         int contador=1;
329         SimpleDateFormat sdf = new SimpleDateFormat("MMM dd,yyyy HH:mm:ss.SSS");
330
331         SortedSet<String> llaveOrden = new TreeSet<>(rs.keySet());
332         for ( String key : llaveOrden ) {
333             try {
334                 //if (util.ISP_MEX(rs.get(key).Path,1)) {
335                     String indice = String.format("%16s", (contador ++)+"/"+rs.size()+".-");
336                     String segmento = String.format("%20s", key);
337                     String nhr=rs.get(key).nextHop;
338                     nhr=util.AcortaIPv6(nhr);
339                     String nextHop = String.format("%20s","("+ nhr+")");  // Para Ipv4
340                     if (nhr.contains(":")) {
341                         segmento = String.format("%25s", key);
342                         nextHop = String.format("%28s","("+ nhr+")");  //  Para Ipv6
343                     }
344                     String Camino = String.format("%75s",rs.get(key).Path);
345                     //System.out.println((contador ++)+"/"+rs.size()+".- "+ padded+"  ("+rs.
                            get(key).nextHop +") "+rs.get(key).Path+"    \t"+SistemasAutonomos.
                            getName("13679"));
346                     System.out.println(indice+" "+ segmento+"  "+nextHop +" "+Camino+" "+sdf.
                            format(new Date(rs.get(key).fecha)));
347
348             } catch (Exception e) {
349                 System.err.println("Error en "+key+" "+e.toString());
350             }
351
352         }
353     }
354
355
356
357     public static void DetalleMexico(Map<String,Ruta> rs) {
358
359         int contador=1;
360         util.LecturaPais("MX");
361         System.out.println("Mostrando AS de Mexico");
362         SortedSet<String> llaveOrden = new TreeSet<>(rs.keySet());
363         for ( String key : llaveOrden ) {
364             try {
365                 String Color;
366                 //if (util.ISP_MEX(rs.get(key).Path,1)) {
367                 if (util.ISP_Pais(rs.get(key).Path,1)) {
368                     String indice = String.format("%16s", (contador ++)+"/"+rs.size()+".-");
369                     String segmento = String.format("%20s", key);
370                     String nextHop = String.format("%20s","("+ rs.get(key).nextHop+")");
371                     //System.out.println((contador ++)+"/"+rs.size()+".- "+ padded+"  ("+rs.
                            get(key).nextHop +") "+rs.get(key).Path+"    \t"+SistemasAutonomos.
                            getName("13679"));
372                     System.out.println(indice+" "+ segmento+" "+ColoresAnsi.PINK+nextHop +"
                            ↪ "+rs.get(key).Path);
373                 }
374
375             } catch (Exception e) {
376                 System.err.println("Error en "+key+" "+e.toString());
377             }
378         }
379     }
380
381     public static void Muestra_Ayuda() {
382         System.out.println("connect <host> - initiate a BGP connection on the default BGP
                ↪ port");
383         System.out.println("status - show status");
384         System.out.println("status <connection> - show routes for that connection");
385         System.out.println("quit - exit, closing everything down.");
386         System.out.println("help or ? - this message");
387         System.out.println("\ninfo - show info");
388         System.out.println("send - Send next hot");
```

```
389             System.out.println("zip - Compact and purge");
390             System.out.println("eva <segment> -Evaluate segment");
391         }
392
393
394
395     }
```

Listado A.12: Reporte.java

```
1   /*
2    * To change this license header, choose License Headers in Project Properties.
3    * To change this template file, choose Tools | Templates
4    * and open the template in the editor.
5    */
6   package bgp.listados;
7
8   import Graficas.Path;
9   import analisis.SistemasAutonomos;
10  import bgp.Ruta;
11  import bgp.util;
12  import java.text.DecimalFormat;
13  import java.util.*;
14
15  /**
16   *
17   * @author lelguea
18   */
19  public class Reporte {
20
21      static int globalc;
22      static int contador;
23
24
25      public static void reporteLatex(Map<String,Ruta> rs) {
26
27          int Indicador_Inicio=Constantes.inicio;
28
29          int minimoI=10;
30          int maximoI=0;
31          String minimoS="";
32          String maximoS="";
33          String maximoRed="";
34          int cuenta=0;
35          int promedio=0;
36
37          Set<String> inicios = new HashSet<>();
38
39          HashMap<String,Integer> Vecinos = new HashMap<>();
40          HashMap<String, Integer> RutasDif = new HashMap<>();
41
42          for ( String key : rs.keySet() ) {
43              if(Vecinos.containsKey(rs.get(key).nextHop)){
44                  int n=Vecinos.get(rs.get(key).nextHop) + 1;
45                  Vecinos.put(rs.get(key).nextHop, n);
46              } else {
47                  Vecinos.put(rs.get(key).nextHop,1);
48              }
49
50              if (!rs.get(key).Path.contains("*")) {
51                  Path p=new Path(rs.get(key).Path);
52
53                  if (p.getSA().size()>Indicador_Inicio) {
54                      inicios.add(""+p.getSA().get(Indicador_Inicio));
55                  }
56
57                  int tam=p.getSA().size();
58                  promedio=promedio+tam;
59                  cuenta=cuenta+1;
60                  if (tam>maximoI) {
```

```
61                      maximoI=tam;
62                      maximoS=rs.get(key).Path;
63                      maximoRed=key;
64                  }
65                  if (tam<minimoI) {
66                      minimoI=tam;
67                      minimoS=rs.get(key).Path;
68                  }
69
70              if(RutasDif.containsKey(rs.get(key).Path)){
71                  int n=RutasDif.get(rs.get(key).Path) + 1;
72                  RutasDif.put(rs.get(key).Path, n);
73              } else {
74                  RutasDif.put(rs.get(key).Path,1);
75              }
76
77              }
78          }
79
80          Map<String, Integer> VecinosO = util.OrdenaMapa(Vecinos);
81
82          DecimalFormat myFormatter = new DecimalFormat("##0.0000");
83
84          System.out.println("\\section{Análisis de Next-Hop}\n");
85          Set nextH = VecinosO.keySet();
86
87          System.out.println("\\begin{table}[h]\n" +
88                             "\\centering\n" +
89                             "\\caption{Diferentes Next-Hop y su utilización}\n" +
90                             "\\begin{tabular}{@{}lll@{}}\n" +
91                             "\\toprule\n" +
92                             "IP                                          & Cantidad & \\%
                                   ↪ \\\\ \\midrule\n");
93
94
95          int suma=0;
96          for (Iterator i = nextH.iterator(); i.hasNext(); ) {
97              String Llave=(String) i.next();
98              String ip=util.AcortaIPv6(Llave);
99              double valor=(100.0*VecinosO.get(Llave))/rs.size();
100             String porc = myFormatter.format(valor);
101             System.out.println(ip+" & "+util.customFormat(VecinosO.get(Llave))+" & "+porc+"\\%
                    ↪ \\\\\n");
102             suma=suma+VecinosO.get(Llave);
103
104         }
105         double valor2=0;
106         if (rs.size()>0) {
107             valor2=(100.0*suma)/rs.size();
108         }
109
110
111         String porc2 = myFormatter.format(valor2);
112         System.out.println(" \\\\ \\midrule \n Suma de Path & "+util.customFormat((suma))+" &
                ↪ "+ porc2 +"\\% \\\\\n Total & "+util.customFormat(rs.size())+" & 100.0000\\%
                ↪ \\\\\n");
113         //System.err.println(" \\\\ \\midrule \n Suma de Path & "+util.customFormat((suma))+"
                ↪ & "+ porc2 +"\\% \\\\\n Total & "+util.customFormat(rs.size())+" & 100.0000\\%
                ↪ \\\\\n");
114
115         System.out.println("\\bottomrule\n" +
116                            "\\end{tabular}\n" +
117                            "\\end{table}");
118
119         //System.out.println("Suma de Path = "+util.customFormat((suma))+"\n\n  Total = "+util.
                customFormat(rs.size())+"\n");
120
121         System.out.println("\\section{Análisis de Path}\n");
122         System.out.println("Se encontraron "+ util.customFormat(RutasDif.size())+" rutas
                ↪ diferentes en los "+util.customFormat(rs.size())+" segmentos de red.\n\n");
123
```

```
124         if (cuenta>0) {
125             System.out.println("Promedio Path= "+myFormatter.format(1.
                    0*promedio/cuenta)+"\n\nMínimo Path  = "+minimoI+" ("+minimoS+")\n\nMáximo
                    ↪ Path  = "+maximoI+" "+maximoRed+".- ("+maximoS+")\n\n");
126         }
127
128         System.out.println("\\section{Conexiones con otros Sistemas Autonomos }\n");
129         System.out.println("Los siguientes sistemas autonomos son parte de los vecinos
                ↪ directos, es decir, "
130                 + "tienen conexión directa con este sistema autonomo. "
131                 + "No debe de aparecer este sistema autonomo en alguna otra parte de la ruta,
                        ↪ a exepción del inicio.\n \n");
132
133         if (inicios.size()>0) {
134         System.out.println(" \\begin{enumerate}\n" );
135         inicios.stream().filter((sa) -> (sa != null )).forEachOrdered((sa) -> {
136             if (sa != null ) {
137                 try {
138                     String nombreAS=SistemasAutonomos.getName(sa);
139                     nombreAS=nombreAS.replaceAll("&","\\\\&");
140                     nombreAS=nombreAS.replaceAll("%","\\\\%");
141                     System.out.println("\\item "+sa+" ("+nombreAS+")\n");
142                 } catch (Exception e) {
143
144                 }
145             }
146         });
147         } else {
148             System.out.println("    \\begin{enumerate}\n" );
149             System.out.println("\\item Sin Sistemas Autonomos\n");
150         }
151         System.out.println(" \\end{enumerate}\n");
152
153
154         globalc=1;
155         System.out.println("\\section{Errores por ISP:}\n");
156         SortedSet<String> llaveOrden = new TreeSet<>(rs.keySet());
157         int respa=1;
158
159         inicios.stream().filter((sa) -> (sa != null )).forEachOrdered((sa) -> {
160             String nombreAS=SistemasAutonomos.getName(sa);
161             nombreAS=nombreAS.replaceAll("&","\\\\&");
162             nombreAS=nombreAS.replaceAll("%","\\\\%");
163             try {
164                 ArrayList<String> rutasOptimas=new ArrayList<>();
165
166                 contador=1;
167                 //for ( String key3 : llaveOrden ) {
168                 llaveOrden.stream().filter((key3) -> (rs.get(key3) !=null)).forEach((key3) -> {
169
170                     try {
171                         //System.out.println("k= "+key3);
172                         //System.out.println("1");
173                         //System.out.println("p= "+rs.get(key3).Path);
174                         if (key3!=null && rs.get(key3).Path !=null && !rs.get(key3).Path.
                                isEmpty()) {
175                             String Camino=rs.get(key3).Path;
176                             int sa_nume=Integer.parseInt(sa);
177                             //System.out.println(Camino+" .- "+sa_nume);
178                             if (util.VecinoEnRuta(Camino, Indicador_Inicio+1, sa_nume)) {
179
180                                 String indice = String.format("%18s", (contador
                                        ↪ ++)+"/"+(globalc++)+"/" +rs.size()+".-");
181                                 String segmento = String.format("%20s", key3);
182                                 String nextHop = String.format("%20s","("+ util.AcortaIPv6(rs.
                                        get(key3).nextHop)+")");
183                                 String formatoPath=rs.get(key3).Path;
184                                 formatoPath=formatoPath.replaceAll(""+sa_
                                        nume,"\\\\textcolor{red}{"+sa_nume+"}");
185                                 rutasOptimas.add(indice+" "+ segmento+"  "+nextHop +"
                                        ↪ "+formatoPath+"\n");
```

```
186                             }
187                         }
188                     } catch (NumberFormatException e) {
189                         //System.err.println("probelma en "+e.toString());
190                         //e.printStackTrace();
191                     }
192                 });
193 
194             if (rutasOptimas.size()>0) {
195                     System.out.println("\n \\section{Buscando errores del AS"+sa+"
                        ↪ ("+nombreAS+")}\n");
196                     System.out.println("\n\\subsection{Rutas no optimizadas}\n");
197                     for (int i=0;i<rutasOptimas.size();i++) {
198                         System.out.println(rutasOptimas.get(i));
199                     }
200                 }
201 
202                 ArrayList<String> privados =new ArrayList<>();
203                 llaveOrden.stream().filter((key3) -> (rs.get(key3) !=null)).forEach((key3) -> {
204                     try {
205                         //System.out.println("k= "+key3);
206                         //System.out.println("1");
207                         //System.out.println("p= "+rs.get(key3).Path);
208 
209                         if (key3!=null && rs.get(key3).Path !=null && !rs.get(key3).Path.
                            isEmpty()) {
210                             if (util.RutaASPrivado(rs.get(key3).Path, Indicador_Inicio,
                                ↪ Integer.parseInt(sa))) {
211 
212                                 String indice = String.format("%18s", (contador
                                    ↪ ++)+"/"+(globalc++)+"/" +rs.size()+".-");
213                                 String segmento = String.format("%20s", key3);
214                                 String nextHop = String.format("%20s","("+ util.AcortaIPv6(rs.
                                    get(key3).nextHop)+")");
215                                 String formatoPath=rs.get(key3).Path;
216                                 for (int asp=64512;asp<=65534;asp++) {
217                                     formatoPath=formatoPath.
                                        replaceAll(""+asp,"\\\\textcolor{green}{"+asp+"}");
218                                 }
219                                 privados.add(indice+" "+ segmento+"  "+nextHop +"
                                    ↪ "+formatoPath+"\n");
220                             }
221                         }
222                     } catch (NumberFormatException ex) {
223                         //System.err.println(ex.toString());
224                     }
225                 });
226 
227             if (privados.size()>0) {
228                     if (rutasOptimas.size()<1) {
229                         System.out.println("\n \\section{Buscando errores del AS"+sa+"
                            ↪ ("+nombreAS+")}\n");
230                     }
231                     System.out.println("\n\\subsection{Rutas con AS Privados}\n");
232                     for (int i=0;i<privados.size();i++) {
233                         System.out.println(privados.get(i));
234                     }
235 
236                 }
237 
238             } catch (NumberFormatException e) {
239                 //System.err.println(e.toString()+"nota: as="+tempoAS);
240             }
241         });
242     }
243 
244 
245 }
```

Listado A.13: RevisaConsola.java

```
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package bgp.listados;
7
8  import Graficas.Path;
9  import analisis.SistemasAutonomos;
10 import bgp.Ruta;
11 import bgp.util;
12 import java.text.DecimalFormat;
13 import java.util.ArrayList;
14 import java.util.HashMap;
15 import java.util.HashSet;
16 import java.util.Iterator;
17 import java.util.Map;
18 import java.util.Set;
19 import java.util.SortedSet;
20 import java.util.TreeSet;
21 import up.varios.ColoresAnsi;
22
23 /**
24  *
25  * @author lelguea
26  */
27 public class RevisaConsola {
28
29     static int globalc;
30     static int contador;
31
32     public static void RevisaN(Map<String,Ruta> rs) {
33
34         int Indicador_Inicio=Constantes.inicio;
35
36         int minimoI=10;
37         int maximoI=0;
38         String minimoS="";
39         String maximoS="";
40         String maximoRed="";
41         int cuenta=0;
42         int promedio=0;
43
44         Set<String> inicios = new HashSet<>();
45         HashMap<String, Integer> RutasDif = new HashMap<>();
46         HashMap<String,Integer> Vecinos = new HashMap<>();
47
48         // Analisis de Path
49         for ( String key : rs.keySet() ) {
50             if(Vecinos.containsKey(rs.get(key).nextHop)){
51                 int n=Vecinos.get(rs.get(key).nextHop) + 1;
52                 Vecinos.put(rs.get(key).nextHop, n);
53             } else {
54                 Vecinos.put(rs.get(key).nextHop,1);
55             }
56
57             if (!rs.get(key).Path.contains("*")) {
58                 Path p=new Path(rs.get(key).Path);
59
60                 if (p.getSA().size()>Indicador_Inicio) {
61                     inicios.add(""+p.getSA().get(Indicador_Inicio));
62                 }
63
64                 int tam=p.getSA().size();
65                 promedio=promedio+tam;
66                 cuenta=cuenta+1;
67                 if (tam>maximoI) {
68                     maximoI=tam;
69                     maximoS=rs.get(key).Path;
70                     maximoRed=key;
71                 }
```

```
72                  if (tam<minimoI) {
73                      minimoI=tam;
74                      minimoS=rs.get(key).Path;
75                  }
76              if(RutasDif.containsKey(rs.get(key).Path)){
77                  int n=RutasDif.get(rs.get(key).Path) + 1;
78                  RutasDif.put(rs.get(key).Path, n);
79              } else {
80                  RutasDif.put(rs.get(key).Path,1);
81              }
82              }
83          }
84
85
86          // Analisis de Nexthop
87          Map<String, Integer> VecinosO = util.OrdenaMapa(Vecinos);
88          DecimalFormat myFormatter = new DecimalFormat("##0.000");
89
90          System.out.println(ColoresAnsi.BLUE+"Análisis de Next-Hop");
91          Set nextH = VecinosO.keySet();
92
93          int suma=0;
94          for (Iterator i = nextH.iterator(); i.hasNext(); ) {
95              String Llave=(String) i.next();
96              double valor=(100.0*VecinosO.get(Llave))/rs.size();
97              String porc = myFormatter.format(valor);
98              System.out.println(Llave+".- "+VecinosO.get(Llave)+" ("+porc+"%)");
99              suma=suma+VecinosO.get(Llave);
100
101         }
102         System.out.println("Suma="+util.customFormat((suma))+" total="+util.customFormat(rs.
                size()));
103
104         System.out.println(ColoresAnsi.BLUE+"Análisis de Path");
105         System.out.println("Se encontraron "+ util.customFormat(RutasDif.size())+" rutas
                ↪ diferentes en los "+util.customFormat(rs.size())+" segmentos de red");
106
107         Map<String, Integer> CaminosO = util.OrdenaMapa(RutasDif);
108
109         Set pathx = CaminosO.keySet();
110
111         // Muestra la cantidad de caminos agrupados por segmentos.
112         for (Iterator i = pathx.iterator(); i.hasNext(); ) {
113             String Camino=(String) i.next();
114             // Reporte Detallado
115             //System.out.println(Camino+" -> "+CaminosO.get(Camino));
116         }
117         if (cuenta>0) {
118             System.out.println("\nPromedio Path= "+myFormatter.format(1.
                    0*promedio/cuenta)+"\nMínimo Path  = "+minimoI+" ("+minimoS+")\nMáximo Path
                    ↪ = "+maximoI+" "+maximoRed+".- ("+maximoS+")");
119         }
120
121         System.out.println(ColoresAnsi.BLUE+"\nLos Sistemas Autónomos inician con:");
122         inicios.stream().filter((sa) -> (sa != null )).forEachOrdered((sa) -> {
123             try {
124                 System.out.println(sa+" ("+SistemasAutonomos.getName(sa)+")");
125             } catch (Exception e) {
126
127             }
128         });
129
130         globalc=1;
131         SortedSet<String> llaveOrden = new TreeSet<>(rs.keySet());
132         //int respa=1;
133         inicios.forEach((sa) -> {
134             try {
135                 ArrayList<String> salidasOptimas=new ArrayList<>();
136
137                 contador=1;
138                 //for ( String key3 : llaveOrden ) {
```

```
139                 llaveOrden.stream().filter((key3) -> (rs.get(key3) !=null)).forEach((key3) -> {
140
141                     if (key3!=null && rs.get(key3).Path !=null && !rs.get(key3).Path.isEmpty())
                            ↪ {
142                         String Camino=rs.get(key3).Path;
143                         int sa_nume=Integer.parseInt(sa);
144                         //System.out.println(Camino+" .- "+sa_nume);
145                         if (util.VecinoEnRuta(Camino, Indicador_Inicio+1, sa_nume)) {
146
147                             String indice = String.format("%18s", (contador
                                    ↪ ++)+"/"+(globalc++)+"/" +rs.size()+".-");
148                             String segmento = String.format("%20s", key3);
149                             String nextHop = String.format("%20s","("+ util.AcortaIPv6(rs.
                                    get(key3).nextHop)+")");
150                             //System.out.println((contador ++)+"/"+rs.size()+".- "+ padded+"
                                    ↪ ("+rs.get(key).nextHop +") "+rs.get(key).Path+"
                                    ↪ \t"+SistemasAutonomos.getName("13679"));
151                             //System.out.println(indice+" "+ segmento+"  "+nextHop +" "+rs.
                                    get(key3).Path);
152                             salidasOptimas.add(indice+" "+ segmento+"  "+nextHop +" "+rs.
                                    get(key3).Path);
153                         }
154                     }
155
156                 });
157                 if (salidasOptimas.size()>0) {
158                     System.out.println(ColoresAnsi.RED+"Errores por Vecino " +sa+"
                            ↪ ("+SistemasAutonomos.getName(sa)+")");
159                     for (int i=0;i<salidasOptimas.size();i++) {
160                         System.out.println(salidasOptimas.get(i));
161                     }
162                 }
163
164                 ArrayList<String> privados =new ArrayList<>();
165                 //llaveOrden.forEach((key3) -> {
166                 llaveOrden.stream().filter((key3) -> (rs.get(key3) !=null)).forEach((key3) -> {
167
168                     if (key3!=null && rs.get(key3).Path !=null && !rs.get(key3).Path.isEmpty())
                            ↪ {
169                         if (util.RutaASPrivado(rs.get(key3).Path, Indicador_Inicio,  Integer.
                                parseInt(sa))) {
170
171                             String indice = String.format("%18s", (contador
                                    ↪ ++)+"/"+(globalc++)+"/" +rs.size()+".-");
172                             String segmento = String.format("%20s", key3);
173                             String nextHop = String.format("%20s","("+util.AcortaIPv6(rs.
                                    get(key3).nextHop)+")");
174                             //System.out.println((contador ++)+"/"+rs.size()+".- "+ padded+"
                                    ↪ ("+rs.get(key).nextHop +") "+rs.get(key).Path+"
                                    ↪ \t"+SistemasAutonomos.getName("13679"));
175                             //System.out.println(indice+" "+ segmento+"  "+nextHop +" "+rs.
                                    get(key3).Path);
176                             privados.add(indice+" "+ segmento+"  "+nextHop +" "+rs.get(key3).
                                    Path);
177                         }
178                     }
179
180                 });
181                 if (privados.size()>0) {
182                     System.out.println(ColoresAnsi.RED+"Rutas con AS Privados de " +sa+"
                            ↪ ("+SistemasAutonomos.getName(sa)+")");
183                     for (int i=0;i<privados.size();i++) {
184                         System.out.println(privados.get(i));
185                     }
186                 }
187
188             } catch (Exception e) {
189                 System.out.println("x"+e.toString());
190
191                 //e.printStackTrace();
```

```
192                     //System.exit(-1);
193                 }
194             });
195 
196 
197 
198         }
199 
200 
201 }
```

Listado A.14: Ruta.java

```
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package bgp;
7 
8  /**
9   *
10  * @author lelguea
11  */
12 public class Ruta  {
13 
14      String IP;
15      public String nextHop;
16      public String Path;
17      public long fecha;
18 
19 
20      public Ruta(String IP, String nextHop, String Path) {
21          this.IP = IP;
22          this.nextHop = nextHop;
23          this.Path = Path;
24          this.fecha=System.currentTimeMillis();
25      }
26 
27      public Ruta(String IP, String nextHop) {
28          this.IP = IP;
29          this.nextHop = nextHop;
30          this.fecha=System.currentTimeMillis();
31      }
32 
33      public String getIP() {
34          return IP;
35      }
36 
37      @Override
38      public String toString() {
39          return "Ruta{" + "IP=" + IP + '}';
40      }
41 
42      public void setIP(String IP) {
43          this.IP = IP;
44      }
45 
46      public String getNextHop() {
47          return nextHop;
48      }
49 
50      public void setNextHop(String nextHop) {
51          this.nextHop = nextHop;
52      }
53 
54      public String getPath() {
55          return Path;
56      }
57 
```

```
58      public void setPath(String Path) {
59          this.Path = Path;
60      }
61
62      public boolean equals(Ruta r ) {
63          return this.IP.equals(r.IP);
64      }
65
66      public boolean named(Ruta r) {
67          return this.IP.equals(r.IP);
68      }
69  }
```

Listado A.15: util.java

```
1   package bgp;
2
3   import Graficas.Path;
4   import analisis.SistemasAutonomos;
5   import db.AdminDB;
6   import java.net.InetAddress;
7   import java.net.UnknownHostException;
8   import java.text.DateFormat;
9   import java.text.DecimalFormat;
10  import java.text.SimpleDateFormat;
11  import java.util.*;
12  //import javax.xml.bind.DatatypeConverter;
13
14  // Se reviso cumplimiento de estandares
15
16  public class util {
17
18      static List<Integer> listado=new ArrayList<>();
19
20      /* container class for miscellaneous util methods */
21
22      public static final void dumpArray(byte b[], int start, int end) {
23          for (int i = 1; i < start % 16; i++) {
24              System.out.print("   ");
25          }
26          int i = start;
27          while (i <= end) {
28              if (i % 8 == 0) System.out.print(" ");
29              if (i % 16 == 0) System.out.println("");
30              System.out.print(Byte.toString(b[i])+" ");
31              i++;
32          }
33          System.out.println("");
34      }
35
36
37      /* java should really make this a tad easier */
38      public static final int intFromBytes(byte b1,  byte b2, byte b3, byte b4) {
39          int i1, i2, i3, i4;
40          i1 = b1; i2 = b2; i3 = b3; i4 = b4;
41          return (((i1 << 24) & 0xff000000) |
42            ((i2 << 16) & 0x00ff0000) |
43         ((i3 << 8) & 0x0000ff00) |
44         (i4 & 0x000000ff));
45      }
46
47      public static final long intFromBytesx(byte b1,  byte b2, byte b3, byte b4) {
48          int i1, i2, i3, i4;
49          i1 = b1; i2 = b2; i3 = b3; i4 = b4;
50          return (((i1 << 24) & 0xff000000) |
51            ((i2 << 16) & 0x00ff0000) |
52         ((i3 << 8) & 0x0000ff00) |
53         (i4 & 0x000000ff));
54      }
```

```
55
56
57
58      public static final long longFromBytes(byte b1, byte b2, byte b3, byte b4, byte b5, byte
            ↪ b6, byte b7, byte b8) {
59          long l1, l2, l3, l4, l5, l6, l7, l8;
60          l1 = b1; l2 = b2; l3 = b3; l4 = b4; l5 = b5; l6 = b6; l7 = b7; l8 = b8;
61          return (((l1 << 56) & 0xff00000000000000L) |
62         ((l2 << 48) & 0x00ff000000000000L) |
63         ((l3 << 40) & 0x0000ff0000000000L) |
64         ((l4 << 32) & 0x000000ff00000000L) |
65         ((l5 << 24) & 0x00000000ff000000L) |
66         ((l6 << 16) & 0x0000000000ff0000L) |
67         ((l7 <<  8) & 0x000000000000ff00L) |
68         (l8 &         0x00000000000000ffL));
69
70          //    l1 = intFromBytes(b8, b7, b6, b5); if (l1 < 0) { l1 = l1 + 65536 ; }
71          //l2 = intFromBytes(b4, b3, b2, b1); if (l2 < 0) { l2 = l2 + 65536 ; }
72          //return (long) ((l1 << 32) | l2);
73      }
74
75      /* get the which'th byte of i, which should be [0..3] with 0 being LSB */
76      public static final byte byteOfInt(int i, int which) {
77          int shift = which * 8;
78          return (byte) ((i & (0x000000FF << shift)) >> shift);
79      }
80
81      public static final byte byteOfLong(long i, int which) {
82          int shift = which * 8;
83          return (byte) (( i & (0x00000000000000FF <<shift)) >> shift);
84      }
85
86      public static final int intOfLong(long i, int whichbyte) {
87          int shift = whichbyte * 8;
88          return (int) (( i & (0x00000000000000FF << shift)) >> shift);
89      }
90
91      /* test routine */
92      public static void main(String argv[]) {
93
94          System.out.println(ASnumtoString32(22566));
95          System.exit(0);
96
97          try {
98              System.out.println(IPv6(InetAddress.getByName("2001:0450:2002:0236::9D")));
99              System.out.println(IPv6(InetAddress.getByName("2001:0450:2002:0236::9D")));
100             System.out.println(IPv6(InetAddress.getByName("2001:450:2002:0236:0::9D")));
101
102             System.out.println(LengNumto1String(32));
103
104             System.exit(0);
105
106             String subjectString="2001:db8:0:0:0:0:2:1";
107             String resultString = subjectString.
                    replaceAll("((?::0\\b){2,}):?(?!\\S*\\b\\1:0\\b)(\\S*)", "::$2");
108             String resultString2 = subjectString.
                    replaceAll("((?:(?:^|:)0+\\b){2,}):?(?!\\S*\\b\\1:0+\\b)(\\S*)", "::$2");
109             System.out.println(resultString+" "+resultString2);
110             System.exit(0);
111
112             byte[] asd=hexStringToByteArray("FDF2");
113             int oasdad=intFromBytes((byte) 0, (byte) 0, asd[0],asd[1]);
114             int oasdad2=(asd[0] << 8) | (asd[1] & 0xFF);
115             System.out.println("FDF2 =" +oasdad);
116             System.out.println("FDF2 =" +oasdad2);
117
118
119
120             System.out.println("0xFFFFFFFF = " +
121                     Integer.toString(intFromBytes((byte) 0xFF, (byte) 0, (byte) 0, (byte) 0))
                            ↪ +
122                     " (should be -2^31)");
```

```
123             System.out.println("0x00000100 = "+ intFromBytes((byte)0, (byte)0, (byte)1,
                    ↪ (byte)0));
124
125             System.out.println(Integer.
                    toString(byteOfInt(intFromBytes((byte)0xFF,(byte)0xFF,(byte)0xFF,
                    ↪ (byte)0xFF),3)));
126             System.out.println(Byte.toString((byte)0xFF));
127
128             System.out.println(ASnumtoString(22566));
129             System.out.println(ASnumtoString(8151));
130             System.out.println(LengNumto1String(12));
131
132             LecturaPais("MX");
133             System.out.println(ISP_Pais("5 6 7 8 9 278 1", 1));
134         } catch (UnknownHostException ex) {
135             System.err.println(ex.toString());
136         }
137
138     }
139
140     public static String ASnumtoString32(int as) {
141         //System.out.println(as+"-"+LengNumto2String(as));
142         return LengNumto4String(as);
143     }
144
145     public static String ASnumtoString(int as) {
146         //System.out.println(as+"-"+LengNumto2String(as));
147         return LengNumto2String(as);
148     }
149
150     public static String LengNumto2String(int num) {
151         byte[] binario = new byte[2];
152         binario[0]=byteOfInt(num, 1);
153         binario[1]=byteOfInt(num, 0);
154         return toHexString(binario);
155     }
156
157     public static String LengNumto4String(int num) {
158         byte[] binario = new byte[4];
159         binario[0]=byteOfInt(num, 3);
160         binario[1]=byteOfInt(num, 2);
161         binario[2]=byteOfInt(num, 1);
162         binario[3]=byteOfInt(num, 0);
163         return toHexString(binario);
164     }
165
166     public static String LengNumto1String(int nume) {
167         byte[] binario = new byte[1];
168         binario[0]=byteOfInt(nume, 0);
169         return toHexString(binario);
170     }
171
172       public static String convertToHexString(byte[] data) {
173         StringBuilder buf = new StringBuilder();
174         for (int i = 0; i < data.length; i++) {
175             int halfbyte = (data[i] >>> 4) & 0x0F;
176             int two_halfs = 0;
177             do {
178                 if ((0 <= halfbyte) && (halfbyte <= 9))
179                     buf.append((char) ('0' + halfbyte));
180                 else
181                     buf.append((char) ('a' + (halfbyte - 10)));
182                 halfbyte = data[i] & 0x0F;
183             } while(two_halfs++ < 1);
184         }
185         return buf.toString();
186     }
187
188     public static byte[] hexStringToByteArray(String s) {
189         int len = s.length();
190         byte[] data = new byte[len / 2];
```

```
191         for (int i = 0; i < len; i += 2) {
192             data[i / 2] = (byte) ((Character.digit(s.charAt(i), 16) << 4)
193                                  + Character.digit(s.charAt(i+1), 16));
194         }
195         return data;
196     }
197 
198     public static String toHexString(byte[] array) {
199 
200         Formatter formatter = new Formatter();
201         for (byte b : array) {
202             formatter.format("%02X", b);
203         }
204 
205         return formatter.toString();
206     }
207 
208     public static byte[] toByteArray(String s) {
209         //return DatatypeConverter.parseHexBinary(s);
210         int l = s.length();
211         byte[] data = new byte[l/2];
212         for (int i = 0; i < l; i += 2) {
213             data[i/2] = (byte) ((Character.digit(s.charAt(i), 16) << 4)
214                                 + Character.digit(s.charAt(i+1), 16));
215         }
216         return data;
217     }
218 
219     static public String customFormat(String pattern, double value ) {
220         DecimalFormat myFormatter = new DecimalFormat(pattern);
221         String output = myFormatter.format(value);
222         return output;
223     }
224 
225     static public String customFormat(double value ) {
226         String pattern="#########.###";
227         DecimalFormat myFormatter = new DecimalFormat(pattern);
228         String output = myFormatter.format(value);
229         return output;
230     }
231 
232     static public String customFormat(int value ) {
233         String pattern="#######";
234         DecimalFormat myFormatter = new DecimalFormat(pattern);
235         String output = myFormatter.format(value);
236         return output;
237     }
238 
239     @Deprecated
240     static public String segmentASs(String str) {
241         byte[] se=util.toByteArray(str);
242         String Salida="";
243 
244         int test=se[0]<<8 &0xFF00 | se[1]&0xFF;
245 
246         if (test!=0) {
247             for (int i=0;i<se.length;i=i+2) {
248                 //int numero=((int)se[i] << 8) | ((int)se[i+1] & 0xFF);
249                 int numero=se[i]<<8 &0xFF00 | se[i+1]&0xFF;
250                 Salida=Salida+" "+numero;
251             }
252         } else {
253             if ((se.length %4)==0) {
254                 // Suponemos que es de 4 bytes los AS, pues no puede haber 0 (ceros)
255                 Salida="";
256                 for (int i=0;i<se.length;i=i+4) {
257                     //int numero=((int)se[i] << 8) | ((int)se[i+1] & 0xFF);
258 
259                     System.out.println(">>l:>>"+se.length+"<<<"+i);
260                     int numero=intFromBytes(se[i],se[i+1],se[i+2],se[i+3]);
261                     //long numerox=longFromBytes((byte) 0,(byte) 0, (byte) 0, (byte) 0,
```

```
                    ↪ se[i],se[i+1],se[i+2],se[i+3]);
262                 Salida=Salida+" "+numero;
263                 }
264             } else {
265                 System.out.println(convertToHexString(se));
266                 for (int i=0;i<se.length;i=i+2) {
267                 //int numero=((int)se[i] << 8) | ((int)se[i+1] & 0xFF);
268                 int numero=se[i]<<8 &0xFF00 | se[i+1] & 0xFF;
269                 Salida=Salida+" "+numero;
270                 }
271                 Salida=Salida+";";
272                 //System.exit(-1);
273             }
274         }
275         return Salida;
276     }
277
278     static public String segmentASs(byte[] se) {
279         String Salida="";
280
281         if (se.length<3) {
282             return "*";
283         }
284
285         int seg_nums=se[1];
286
287
288         if (seg_nums*2==se.length-2 && se[3]!=0) {   //   Procesamos si es solo el Path, sin
                ↪ nada adicional.... y 16 bits x AS
289             for (int i=2;i<se.length;i=i+2) {
290                 //int numero=((int)se[i] << 8) | ((int)se[i+1] & 0xFF);
291                 int numero=intFromBytes((byte) 0, (byte) 0,se[i],se[i+1]);
292                 Salida=Salida+" "+numero;
293             }
294         } else {
295             String Segunda=util.toHexString(se).substring(seg_nums*4+4,seg_nums*4+6);
296             //System.out.println("Agregado0 "+util.toHexString(se).substring(seg_nums*4+4,seg_
                    nums*4+8));
297             if ((Segunda.startsWith("01") || Segunda.startsWith("02")) && se[3]!=0 ) {  //
                    ↪ Procesamos y anadimos al as-set=1  (Agregregado)  o as-set=2
298                 // en un futuro se puede añadir { y } entre el agregado....
299                 // https://www.cisco.com/c/es_mx/support/docs/ip/border-gateway-protocol-
                        bgp/5441-aggregation.html
300                         //System.out.println("ok o no "+(seg_nums*2)+" "+(se.length-2));
301                 //System.out.println("Cant= "+seg_nums+"  "+ util.toHexString(se).
                        substring(4)+" "+(se.length-2));
302                 //System.out.println("Agregado1 "+util.toHexString(se).substring(seg_
                        nums*4+4,seg_nums*4+8));
303
304                 for (int i=2;i<seg_nums*2+2;i=i+2) {
305                 //int numero=((int)se[i] << 8) | ((int)se[i+1] & 0xFF);
306                 int numero=intFromBytes((byte) 0, (byte) 0,se[i],se[i+1]);
307                 Salida=Salida+" "+numero;
308                 }
309
310                 for (int i=seg_nums*2+4;i<se.length;i=i+2) {
311                    int numero=intFromBytes((byte) 0, (byte) 0,se[i],se[i+1]);
312                 Salida=Salida+" "+numero;
313                 }
314                 //System.out.println(Salida);
315
316             } else {
317
318                 if (seg_nums*4==(se.length-2)) { // Coincide la información para AS de 32 bits...
319                     for (int i=2;i<se.length;i=i+4) {
320                         //int numero=((int)se[i] << 8) | ((int)se[i+1] & 0xFF);
321                         int numero=intFromBytes(se[i],se[i+1],se[i+2],se[i+3]);
322                         long numerox=longFromBytes((byte) 0,(byte) 0, (byte) 0, (byte) 0,
                                ↪ se[i],se[i+1],se[i+2],se[i+3]);
323                         Salida=Salida+" "+numerox;
324                     }
```

```
325                 } else {
326                     //System.out.println("Cant32= "+seg_nums+"  "+ util.toHexString(se).
                            substring(4)+" "+(se.length-2));
327                     String Segunda32=util.toHexString(se).substring(seg_nums*8+4,seg_nums*8+6);
328                     //System.out.println("Segunda32... ="+util.toHexString(se).substring(seg_
                            nums*8+4,seg_nums*8+8));
329                     if (Segunda32.startsWith("01") || Segunda32.startsWith("02")) {
330                         int siguiente=(int) se[seg_nums*4+3];
331                         //System.out.println("siguiente ="+siguiente);
332                         for (int i=2;i<seg_nums*4+2;i=i+4) {
333                             //int numero=((int)se[i] << 8) | ((int)se[i+1] & 0xFF);
334                             int numero=intFromBytes(se[i],se[i+1],se[i+2],se[i+3]);
335                             long numerox=longFromBytes((byte) 0,(byte) 0, (byte) 0, (byte) 0,
                                    ↪ se[i],se[i+1],se[i+2],se[i+3]);
336                             Salida=Salida+" "+numerox;
337                         }
338 
339                         for (int i=seg_nums*4+4;i<seg_nums*4+4+siguiente*4;i=i+4) {
340                             int numero=intFromBytes(se[i],se[i+1],se[i+2],se[i+3]);
341                             long numerox=longFromBytes((byte) 0,(byte) 0, (byte) 0, (byte) 0,
                                    ↪ se[i],se[i+1],se[i+2],se[i+3]);
342                             Salida=Salida+" "+numerox;
343                         }
344 
345                     } else {
346                         System.out.println("NA"+util.toHexString(se));
347                         Salida="+-+";
348                     }
349 
350                 }
351             }
352 
353         }
354 
355 
356         return Salida;
357     }
358 
359         public static boolean ISP_MEX(String Camino) {
360         boolean Salida=false;
361 
362         ArrayList<Integer> ali=new Path(Camino).getSA();
363         for (int i=0;i<ali.size();i++) {
364             if (SistemasAutonomos.getPais(""+ali.get(i)).equalsIgnoreCase("MX")) {
365                 Salida=true;
366             }
367         }
368 
369         return Salida;
370     }
371 
372     public static boolean ISP_MEX(String Camino, int inicio) {
373         boolean Salida=false;
374 
375         try {
376 
377         ArrayList<Integer> ali=new Path(Camino).getSA();
378         if (inicio==-1) {
379             inicio=ali.size()-1;
380         }
381         if (inicio>=ali.size()) {
382             return false;
383         }
384         for (int i=inicio;i<ali.size();i++) {
385             if (SistemasAutonomos.getPais(""+ali.get(i)).equalsIgnoreCase("MX")) {
386                 Salida=true;
387                 return Salida;
388             }
389         }
390         } catch (Exception e) {
391 
392         }
```

```
393
394         return Salida;
395     }
396
397     public static boolean RutaASPrivado(String Camino, int inicio, int ISP) {
398         boolean Salida=false;
399
400         try {
401
402         ArrayList<Integer> ali=new Path(Camino).getSA();
403         for (int i=0;i<ali.size();i++) {
404             //if (ali.get(i)==ISP ) {    Esto puede presentar errores en caso de prepend del
                    ↪ mismos vecino
405             if (ali.get(inicio)==ISP && ali.get(i)>=64512 && ali.get(i)<=65534) {  // Asi se
                    ↪ revisa que no sea el mimso veciono el que se anuncia en 2 camino...
406
407                 Salida=true;
408                 return Salida;
409             }
410         }
411
412         } catch (Exception e) {
413
414         }
415
416         return Salida;
417     }
418
419
420
421     public static boolean VecinoEnRuta(String Camino, int inicio, int ISP) {
422         boolean Salida=false;
423
424         try {
425         //System.out.println("sa="+ISP+" key3="+inicio+" path="+Camino);
426
427         ArrayList<Integer> ali=new Path(Camino).getSA();
428         if (inicio==-1) {
429             inicio=ali.size()-1;
430         }
431         if (inicio>=ali.size()) {
432             return false;
433         }
434         for (int i=inicio;i<ali.size();i++) {
435             //if (ali.get(i)==ISP ) {    Esto puede presentar errores en caso de prepend del
                    ↪ mismos vecino
436             if (ali.get(inicio-1)!=ISP && ali.get(i)==ISP ) {  // Asi se revisa que no sea el
                    ↪ mimso veciono el que se anuncia en 2 camino...
437                 Salida=true;
438                 return Salida;
439             }
440         }
441
442         } catch (Exception e) {
443             System.err.println(e.toString());
444             //e.printStackTrace();
445             System.exit(-1);
446
447         }
448
449         return Salida;
450     }
451
452     public static void LecturaPais(String Pais) {
453         AdminDB db= new AdminDB();
454         db.conecta();
455         listado = db.LecturaSAPais(Pais);
456         db.cierra();
457     }
458
459     public static boolean ISP_Directo(String Camino) {
460         ArrayList<Integer> ali=new Path(Camino).getSA();
```

```
461         return ali.size()==1;
462     }
463
464
465     public static boolean ISP_Pais(String Camino, int inicio) {
466         boolean Salida=false;
467
468
469         ArrayList<Integer> ali=new Path(Camino).getSA();
470         if (inicio==-1) {
471             inicio=ali.size()-1;
472         }
473         if (inicio>=ali.size()) {
474             return false;
475         }
476         for (int i=inicio;i<ali.size();i++) {
477             if (listado.contains(ali.get(i)) ) {
478                 Salida=true;
479                 return Salida;
480             }
481         }
482
483         return Salida;
484     }
485
486     public static String formatoNum(double x) {
487         DecimalFormat miformato = new DecimalFormat("##0.000");
488         return miformato.format(x);
489     }
490
491     public static String formatoNum2(double x) {
492         DecimalFormat miformato = new DecimalFormat("#,##0");
493         return miformato.format(x);
494     }
495
496     public static String Actividad(long elapsed) {
497         DateFormat df = new SimpleDateFormat("HH 'hours', mm 'mins,' ss 'seconds'");
498         DecimalFormat DIA = new DecimalFormat("#00");
499         df.setTimeZone(TimeZone.getTimeZone("GMT+0"));
500         double dia=Math.floor(1.0*elapsed/1000/3600/24);
501         return DIA.format(dia)+" days, "+ df.format(new Date(elapsed));
502     }
503
504     public static boolean esIpv6(String ip) {
505         boolean Salida;
506
507         int pri=ip.indexOf(":")+1;
508         Salida=(ip.indexOf(":", pri)>0);
509
510         return Salida;
511     }
512
513     public static String AcortaIPv6(String ip) {
514
515         //if (ip.contains("::")) {
516         //    return ip;
517         //}
518
519         ip=ip.replaceAll(":0000", ":0");
520         ip=ip.replaceAll(":000", ":0");
521         ip=ip.replaceAll(":00", ":0");
522         //ip=ip.replaceAll(":0:", ":x:");
523         //ip=ip.replaceAll(":0[^0]", ":");
524
525         //ip=ip.replaceAll(":x:", ":0:");
526
527         String resultString = ip.replaceAll("((?::0\\b){2,}):?(?!\\S*\\b\\1:0\\b)(\\S*)",
                ↪ "::$2");
528
529         //resultString=resultString.replaceAll(":0000::", "::");
530         resultString=resultString.replaceAll(":::", "::");
531         resultString=resultString.replaceAll(":0:", ":X:");
```

```
532         resultString=resultString.replaceAll(":0:", ":X:");
533         resultString=resultString.replaceAll(":0:", ":X:");
534         resultString=resultString.replaceAll(":0", ":");
535         resultString=resultString.replaceAll(":X", ":0");
536         resultString=resultString.replaceAll("0000::",":");
537         resultString=resultString.replaceAll(":0::",":");
538
539
540         return resultString;
541         //return ip;
542     }
543
544     public static HashMap OrdenaMapa(HashMap map) {
545         List list = new LinkedList(map.entrySet());
546         // Defined Custom Comparator here
547         Collections.sort(list, (Object o1, Object o2) -> ((Comparable) ((Map.Entry) (o1)).
                getValue()).
548                 compareTo(((Map.Entry) (o2)).getValue()));
549
550        // Here I am copying the sorted list in HashMap
551        // using LinkedHashMap to preserve the insertion order
552        HashMap sortedHashMap = new LinkedHashMap();
553        for (Iterator it = list.iterator(); it.hasNext();) {
554               Map.Entry entry = (Map.Entry) it.next();
555               sortedHashMap.put(entry.getKey(), entry.getValue());
556        }
557        return sortedHashMap;
558     }
559
560     public static HashMap OrdenaMapaD(HashMap map) {
561         List list = new LinkedList(map.entrySet());
562         /* Formato Antiguo
563         Collections.sort(list, new Comparator() {
564             @Override
565             public int compare(Object o2, Object o1) {
566                 return ((Comparable) ((Map.Entry) (o1)).getValue()).
567                 compareTo(((Map.Entry) (o2)).getValue());
568             }
569         });
570         */
571
572         // Defined Custom Comparator here
573         Collections.sort(list, (Object o2, Object o1) -> ((Comparable) ((Map.Entry) (o1)).
                getValue()).
574                 compareTo(((Map.Entry) (o2)).getValue()));
575
576        // Here I am copying the sorted list in HashMap
577        // using LinkedHashMap to preserve the insertion order
578        HashMap sortedHashMap = new LinkedHashMap();
579        for (Iterator it = list.iterator(); it.hasNext();) {
580               Map.Entry entry = (Map.Entry) it.next();
581               sortedHashMap.put(entry.getKey(), entry.getValue());
582        }
583        return sortedHashMap;
584     }
585
586     public static String IPv6(InetAddress dir) {
587
588
589         String Salida;
590         byte[] b = dir.getAddress();
591
592         Salida=util.toHexString(b);
593
594
595         return Salida;
596     }
597
598     // Funcion para recortar los bytes necesarios para los anuncios de los segmentos de IPv6
599     // Ejemplo: System.out.println(Segmento(Inet6Address.
            getByName("2801:f0:20:c20:172:25:150:19"),32));
```

```
600     // Ejemplo: System.out.println(Segmento(Inet6Address.
            getByName("2801:f0:20:c20:172:25:150:19"),48));
601
602     public static String Segmento(InetAddress dir, int segmento) {
603         String sip6=util.toHexString(dir.getAddress());
604         int byt=(segmento+7)/8;
605         return sip6.substring(0,2*byt);
606     }
607
608     public static int TamRed(String ip) {
609
610         if (ip.contains("/")) {
611             return Integer.parseInt(ip.substring(ip.indexOf("/")+1));
612         } else if (ip.contains(":")) {
613             return 128;
614         } else {
615             return 32;
616         }
617
618     }
619
620
621 }
```

A.3. Código Java que implementa diferentes REST API para ODL

Listado A.16: GetPutDefault.java

```
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package ODL.bgp_server;
7
8  import java.io.BufferedReader;
9  import java.io.DataOutputStream;
10 import java.io.IOException;
11 import java.io.InputStreamReader;
12 import java.net.HttpURLConnection;
13 import java.net.URL;
14 import java.util.Base64;
15
16
17 /**
18  *
19  * @author lelguea
20  */
21 public class GetPutDefault {
22
23     static String Server="localhost";
24     static String idODLBGP="ejemplo-up";
25     static String appProgRib="192.168.5.224";
26
27     public static String getHTML(String urlToRead) {
28         String Salida="";
29         StringBuilder result = new StringBuilder();
30         try {
31
32             String encoding = Base64.getEncoder().encodeToString(
33       ("admin:admin").getBytes());
34
35             System.out.println(encoding);
36             URL url = new URL(urlToRead);
37             HttpURLConnection conn = (HttpURLConnection) url.openConnection();
38
39
40             conn.setRequestMethod("GET");
41             conn.setRequestProperty("Authorization", "Basic " + encoding);
```

```
42              conn.setRequestProperty("Accept", "text/html,application/xhtml+xml,"
43                      + "application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8");
44              try (BufferedReader rd = new BufferedReader(
45        new InputStreamReader(conn.getInputStream()))) {
46                  int line;
47                  char c;
48                  int pagina=0;
49                  while ((line = rd.read()) != -1) {
50                      pagina++;
51                      c=(char)  line;
52                      //result.append(c);
53                      System.out.print(c);
54                      if (pagina>240) {
55                          System.out.println();
56                          //System.out.println( Runtime.getRuntime().maxMemory()+" "
57          //+ Runtime.getRuntime().totalMemory()+" "
58          //+ Runtime.getRuntime().freeMemory());
59                          pagina=0;
60                      }
61                  }
62              }
63              Salida= result.toString();
64          } catch (IOException e)  {
65              System.out.println(e.toString());
66          }
67          System.out.println(" \n");
68          return Salida;
69
70      }
71
72      private static void send(String Url, String Parametros, String Metodo)
73           throws Exception {
74
75          String encoding = Base64.getEncoder().encodeToString(
76                  ("admin:admin").getBytes());
77
78          URL url = new URL(Url);
79          HttpURLConnection conn = (HttpURLConnection) url.openConnection();
80          //OutputStream os = conn.getOutputStream();
81          //add reuqest header
82          conn.setDoOutput(true);
83          conn.setInstanceFollowRedirects(false);
84          conn.setRequestMethod(Metodo);
85          //con.setRequestProperty("User-Agent", USER_AGENT);
86          conn.setRequestProperty("Accept-Language", "en-US,en;q=0.5");
87          conn.setRequestProperty("Authorization", "Basic " + encoding);
88          conn.setRequestProperty("Content-Type", "application/xml");
89
90
91          try ( // Send post request
92          //conn.setDoOutput(true);
93              DataOutputStream wr = new DataOutputStream(conn.getOutputStream())) {
94              wr.writeBytes(Parametros);
95              wr.flush();
96          }
97
98          int responseCode = conn.getResponseCode();
99          System.out.println("\nSending '"+Metodo+ "' request to URL : " + url);
100         System.out.println("Post parameters : " + Parametros);
101         System.out.println("Response Code : " + responseCode);
102
103         StringBuffer response;
104         try (BufferedReader in = new BufferedReader(
105                 new InputStreamReader(conn.getInputStream()))) {
106             String inputLine;
107             response = new StringBuffer();
108             while ((inputLine = in.readLine()) != null) {
109                 response.append(inputLine);
110             }
111         }
112
```

```
113         System.out.println(response.toString());
114     }
115
116     private static void sendPut(String Url, String Parametros) throws Exception {
117         send(Url,Parametros,"PUT");
118     }
119
120     private static void sendPost(String Url, String Parametros) throws Exception {
121         send(Url,Parametros,"POST");
122     }
123
124
125     private static void sendDelete(String Url) throws Exception {
126         send(Url,"","DELETE");
127     }
128
129
130     public static void main(String[] args) throws Exception {
131         getHTML("http://"+Server+ ":8181/restconf/config/odl-bgp-peer-acceptor"
132                 + "-config:bgp-peer-acceptor-config/default");
133
134         /*
135         sendPut("http://"+Server+ ":8181/restconf/config/odl-bgp-peer-acceptor-
136         config:bgp-peer-acceptor-config/default",
137                 "<bgp-peer-acceptor-config xmlns=\"urn:opendaylight:params:
138         xml:ns:yang:odl-bgp-peer-acceptor-config\">"
139                         + "<config-name>default</config-name><binding-address>
140         0.0.0.0</binding-address>"
141                         + "<binding-port>179</binding-port></bgp-peer-
142         acceptor-config>");
143
144         System.exit(0);
145         */
146
147         //configure the Application Peer:
148         // Solo una vez
149
150         /*
151         sendPost("http://"+Server+ ":8181/restconf/config/openconfig-network-
152         instance:network-instances/network-instance/global-bgp/openconfig-
153         network-instance:protocols/protocol/openconfig-policy-types:BGP/"
154         +idODLBGP+"/bgp/neighbors",
155                 "<neighbor xmlns=\"urn:opendaylight:params:xml:ns:yang:bgp
156         :openconfig-extensions\">\n" +
157                 "    <neighbor-address>"+appProgRib+"</neighbor-address>\n" +
158                 "    <config>\n" +
159                 "        <peer-group>application-peers</peer-group>\n" +
160                 "    </config>\n" +
161                 "</neighbor>");
162         */
163
164         // inject a route into the programmable RIB.
165         String ruta="192.100.172.0/24";
166         String nh="192.168.5.92";
167
168
169         sendPost("http://"+Server+ ":8181/restconf/config/bgp-rib:"
170                 + "application-rib/"+appProgRib+"/tables/bgp-types:"
171                         + "ipv4-address-family/bgp-types:"
172                         + "unicast-subsequent-address-family/bgp-inet:"
173                         + "ipv4-routes",
174                 "<ipv4-route xmlns=\"urn:opendaylight:params:xml:ns:"
175                         + "yang:bgp-inet\">"+
176                 "<path-id>0</path-id>"+
177                 "<prefix>10.0.0.11/32</prefix>"+
178                 "    <attributes>"+
179                 "        <as-path></as-path>"+
180                 "        <origin>"+
181                 "            <value>igp</value>"+
182                 "        </origin>"+
183                 "        <local-pref>"+
```

```
184                     "              <pref>100</pref>"+
185                     "          </local-pref>"+
186                     "          <ipv4-next-hop>"+
187                     "              <global>10.11.1.1</global>"+
188                     "          </ipv4-next-hop>"+
189                     "      </attributes>"+
190                     "</ipv4-route>"
191         );
192             //sendDelete("http://"+Server+ ":8181/restconf/config/bgp-rib:
193             //application-rib/"+appProgRib+"/tables/bgp-types:ipv4-address-family/
194             //bgp-types:unicast-subsequent-address-family/bgp-inet:ipv4-routes/
195             //ipv4-route/"+codifica(ruta)+"/0");
196             //sendDelete("http://"+Server+ ":8181/restconf/config/bgp-rib:
197             //application-rib/"+appProgRib+"/tables/bgp-types:ipv4-address-family/
198             //bgp-types:unicast-subsequent-address-family/bgp-inet:ipv4-routes/
199             //ipv4-route/10.0.0.11%2F32/0");
200
201         }
202
203         public static String codifica(String param) {
204             String Salida=param.replaceAll("/", "%2F");
205
206             return Salida;
207
208         }
209
210     }
```

A.4. Código Java Ping para diferentes proveedores de Internet

Listado A.17: Latencia.java

```
1   package latencia;
2
3   import java.io.*;
4   import java.text.SimpleDateFormat;
5   import java.util.Date;
6   import java.util.Properties;
7   import org.apache.commons.net.telnet.TelnetClient;
8
9   /**
10   *
11   * @author lelguea
12   */
13  public class Latencia {
14
15      private final TelnetClient telnet = new TelnetClient();
16      //private BufferedReader reader;
17      private InputStream in;
18      private PrintStream out;
19      private String serverName;
20      private String user;
21      private String password;
22      private String equipo;
23      public static String promptComplete;
24      public static String promptCompleteEna;
25      public static String promptCompleteP;
26      public static String promptCompleteConf;
27      public static String promptCompleteACL;
28
29      public void connect() throws IOException {
30          String axtel;
31          String bestel;
32          String maxcom;
33
34          String destino="www.google.com";
35          int Cant=10;
36          //int tamanio=(int) (800 + Math.random() * 400);
37          int tamanio=800;
```

```
38          serverName = "ipv6_mix";
39          user= "XXXXXXXX";
40          password = "XXXXXXXX";
41          equipo="00A-A";
42          promptComplete=equipo+">";
43          promptCompleteEna=equipo+"#";
44          promptCompleteP=equipo+"#";
45          promptCompleteConf=equipo+"(config)#";
46          promptCompleteACL=equipo+"(config-ipv6-acl)#";
47          telnet.connect(serverName, 23);
48          in = telnet.getInputStream();
49          out = new PrintStream(telnet.getOutputStream());
50          readUntil("Username: ");
51          write(user);
52          readUntil("Password: ");
53          write(password);
54          readUntil(promptComplete);
55          write("enable");
56          readUntil("Password: ");
57          write("XXXXXXXX");
58          readUntil(promptCompleteEna);
59          write("terminal len 0");
60          readUntil(promptCompleteEna);
61          write("ping "+destino+" size "+tamanio+" source GigabitEthernet0/0 repeat "+Cant);
62          axtel=readUntil(promptCompleteP);
63          write("ping "+destino+" size "+tamanio+" source GigabitEthernet0/1.8 repeat "+Cant);
64          maxcom=readUntil(promptCompleteP);
65          write("ping "+destino+" size "+tamanio+" source GigabitEthernet0/1.10 repeat "+Cant);
66          bestel=readUntil(promptCompleteP);
67
68          SimpleDateFormat dt = new SimpleDateFormat("yyyy/MM/dd hh:mm:ss a");
69
70          axtel=axtel.replaceAll("Success rate is ", "");
71          bestel=bestel.replaceAll("Success rate is ", "");
72          maxcom=maxcom.replaceAll("Success rate is ", "");
73
74          axtel=axtel.replaceAll(" round-trip min/avg/max =", "");
75          bestel=bestel.replaceAll(" round-trip min/avg/max =", "");
76          maxcom=maxcom.replaceAll(" round-trip min/avg/max =", "");
77
78          axtel=axtel.replaceAll(" percent ", ",");
79          bestel=bestel.replaceAll(" percent ", ",");
80          maxcom=maxcom.replaceAll(" percent ", ",");
81
82          axtel=axtel.replaceAll("/", ",");
83          bestel=bestel.replaceAll("/", ",");
84          maxcom=maxcom.replaceAll("/", ",");
85
86          axtel=axtel.replaceAll(" ms", ",");
87          bestel=bestel.replaceAll(" ms", ",");
88          maxcom=maxcom.replaceAll(" ms", ",");
89
90          axtel=axtel.replaceAll("\\(", "");
91          bestel=bestel.replaceAll("\\(", "");
92          maxcom=maxcom.replaceAll("\\(", "");
93          axtel=axtel.replaceAll("\\)", "");
94          bestel=bestel.replaceAll("\\)", "");
95          maxcom=maxcom.replaceAll("\\)", "");
96
97          String[] corta=axtel.split("\\r?\\n");
98          System.out.println("a, "+corta[corta.length-2]+dt.format(new Date()));
99
100         //System.out.println(axtel);
101         //System.out.println(bestel);
102         //System.out.println(maxcom);
103
104         corta=bestel.split("\\r?\\n");
105         System.out.println("b, "+corta[corta.length-2]+dt.format(new Date()));
106
107         corta=maxcom.split("\\r?\\n");
108         System.out.println("m, "+corta[corta.length-2]+dt.format(new Date()));
```

```
109
110     }
111
112     public String readUntil(String pattern) {
113         StringBuilder sb = new StringBuilder();
114
115         try {
116             char lastChar = pattern.charAt(pattern.length() - 1);
117
118             //boolean found = false;
119
120             int check = in.read();
121             char ch = (char) check;
122             while (check != -1) {
123                 //System.out.print(ch);
124                 sb.append(ch);
125                 if (ch == lastChar) {
126                     if (sb.toString().endsWith(pattern)) {
127
128                         return sb.toString();
129                     }
130                 }
131                 check = in.read();
132                 ch = (char) check;
133             }
134         } catch (IOException e) {
135              System.err.println("readUntil exception: "+e.toString());
136         }
137
138         return sb.toString();
139     }
140
141     public void write(String value) {
142         try {
143             out.println(value);
144             out.flush();
145             //System.out.println(value);
146         } catch (Exception e) {
147             System.err.println("write exception: "+e.toString());
148         }
149     }
150
151     public static void main(String[] args) throws IOException {
152         Properties defaultProps = System.getProperties(); //obtiene las "properties" del
                ↪ sistema
153         defaultProps.put("java.net.preferIPv6Addresses", "true");//mapea el valor true en la
                ↪ variable java.net.preferIPv6Addresses
154
155
156         Latencia apli = new Latencia();
157         apli.connect();
158     }
159
160 }
```

Apéndice B

Resultados detallados del análisis de Rutas y Características del hardware de red utilizado

B.1. Router Universidad Panamericana

El ruteador de la Universidad Panamericana es un CISCO2921/K9 con las siguientes características:

- Cisco CISCO2921/K9 (revision 1.0) with 2506752K/114688K bytes of memory.
- Processor board ID FTX1634AJN0.
- 4 Gigabit Ethernet interfaces.
- 1 terminal line.
- 1 Virtual Private Network (VPN) Module.
- DRAM configuration is 64 bits wide with parity enabled.
- 255K bytes of non-volatile configuration memory.
- 250880K bytes of ATA System CompactFlash 0 (Read/Write).

Este equipo es utilizado para alojar el Sistema Autónomo con identificación AS13679 y tiene los siguientes sistemas autónomos vecinos:

- AS6503 propiedad de Axtel, S.A.B. de C.V.
- AS18734 propiedad de Operbes, S.A. de C.V.
- AS22566 propiedad de Maxcom Telecomunicaciones, S.A.B. de C.V.

Con todos éstos ASs los enlaces utilizan indistintamente los protocolos IPv4 e IPv6.

B.1.1. Análisis de rutas IPv4 desde la Universidad Panamericana 2019

Este es el resultado de analizar rutas de IPv4 el 29/11/2019

Listado B.1: Resultados rutas UP IPv4 2019

```
Now listening on PC-14665:179...
>Versión 4
BGP ID 192.100.201.200/32
# Parámetros 5
received OPEN
Conect to /172.25.150.19 from /192.100.201.200:27518
AS = 13679
```

```
Connection not a dup; proceeding.
>st
Mostrando 192.100.201.200/32
192.100.201.200/32 up for 00 days, 00 hours, 00 mins, 07 seconds.
Updates: 42397  Announces: 276001  Witdrawls: 0  Routes: 276013
>st
Mostrando 192.100.201.200/32
192.100.201.200/32 up for 00 days, 00 hours, 00 mins, 15 seconds.
Updates: 88813  Announces: 553963  Witdrawls: 0  Routes: 553963
>st
Mostrando 192.100.201.200/32
192.100.201.200/32 up for 00 days, 00 hours, 00 mins, 27 seconds.
Updates: 110587  Announces: 681055  Witdrawls: 0  Routes: 680987
>st
Mostrando 192.100.201.200/32
192.100.201.200/32 up for 00 days, 00 hours, 00 mins, 32 seconds.
Updates: 110593  Announces: 681060  Witdrawls: 1  Routes: 680988
>revisa
Conexiones de 192.100.201.200/32 is:
Análisis de Next-Hop
192.100.230.254/32.- 1 (0.000%)
192.100.201.200/32.- 3 (0.000%)
201.161.34.97/32.- 76281 (11.202%)
148.245.154.1/32.- 175103 (25.713%)
189.202.194.198/32.- 429599 (63.085%)
Suma=680,987 total=680,987
Análisis de Path
Promedio Path= 4.108
Mínimo Path  = 1 (22566)
Máximo Path  = 34 106.76.160.0/20.- (6503 6453 55644 55644 55644 55644
55644 55644 55644 55644 55644 55644 55644 55644 55644 55644 55644 55644
55644 55644 55644 55644 55644 55644 55644 55644 55644 55644 55644 55644
55644 55644 55644 45271)
Los Sistemas Autónomos inician con:
6503 (Axtel, S.A.B. de C.V., MX)
18734 (Operbes, S.A. de C.V., MX)
22566 (Maxcom Telecomunicaciones, S.A.B. de C.V., MX)
Buscando Errores por ISP
6503 (Axtel, S.A.B. de C.V., MX)
Rutas no optimizadas:
1/1/680993.-        132.247.161.0/24       (189.202.194.198/32) 18734 174 6503 28400 278
2/2/680993.-        132.247.196.0/24       (189.202.194.198/32) 18734 174 6503 28400 278
3/3/680993.-        132.247.58.0/24        (189.202.194.198/32) 18734 174 6503 28400 278
4/4/680993.-        132.247.63.0/24        (189.202.194.198/32) 18734 174 6503 28400 278
5/5/680993.-        132.247.65.0/24        (189.202.194.198/32) 18734 174 6503 28400 278
6/6/680993.-        132.247.88.0/24        (189.202.194.198/32) 18734 174 6503 28400 278
7/7/680993.-        132.247.89.0/24        (189.202.194.198/32) 18734 174 6503 28400 278
8/8/680993.-        132.248.142.0/24       (189.202.194.198/32) 18734 174 6503 28400 278
9/9/680993.-        132.248.179.0/24       (189.202.194.198/32) 18734 174 6503 28400 278
10/10/680993.-      132.248.185.0/24       (189.202.194.198/32) 18734 174 6503 28400 278
11/11/680993.-      132.248.219.0/24       (189.202.194.198/32) 18734 174 6503 28400 278
12/12/680993.-      132.248.227.0/24       (189.202.194.198/32) 18734 174 6503 28400 278
13/13/680993.-      132.248.248.0/24       (189.202.194.198/32) 18734 174 6503 28400 278
14/14/680993.-      132.248.30.0/24        (189.202.194.198/32) 18734 174 6503 28400 278
15/15/680993.-      148.206.147.0/24       (189.202.194.198/32) 18734 174 6503 28400 3596
18734 (Operbes, S.A. de C.V., MX)
Rutas no optimizadas:
22566 (Maxcom Telecomunicaciones, S.A.B. de C.V., MX)
Rutas no optimizadas:
>
>quit
Quiting...
Quiting OK
```

B.1.2. Análisis de rutas IPv6 desde la Universidad Panamericana 2019

Este es el resultado de analizar rutas de IPv6 el 29/11/2019

Listado B.2: Resultados rutas UP IPv6 2019

```
Now listening on lap-19643:179...
>connect 2001:1238::3:2
>Conexión IPv6
Versión 4
BGP ID 192.100.201.200/32
# Parámetros 5
received OPEN
Conect to /2001:0:53aa:64c:c62:78f8:4233:bab9 from /2001:1238:0:0:0:0:3:2:179
AS = 13679
Connection not a dup; proceeding.
Keepalive sent
st
Mostrando 192.100.201.200/32
192.100.201.200/32 up for 00 days, 00 hours, 00 mins, 03 seconds.
Updates: 10303  Announces: 23273  Witdrawls: 0  Routes: 23273
>st
Mostrando 192.100.201.200/32
192.100.201.200/32 up for 00 days, 00 hours, 00 mins, 13 seconds.
Updates: 19547  Announces: 45926  Witdrawls: 1  Routes: 45870
>st
Mostrando 192.100.201.200/32
192.100.201.200/32 up for 00 days, 00 hours, 00 mins, 17 seconds.
Updates: 19560  Announces: 45941  Witdrawls: 1  Routes: 45871
>st
Mostrando 192.100.201.200/32
192.100.201.200/32 up for 00 days, 00 hours, 00 mins, 19 seconds.
Updates: 19566  Announces: 45948  Witdrawls: 1  Routes: 45871
>revisa
Conexiones de 192.100.201.200/32 is:
Análisis de Next-Hop
2001:1238:0000:0000:0000:0000:0003:0002.- 2 (0.004%)
2001:1278:0001:0000:0000:0000:0000:00f2.- 12491 (27.231%)
2001:1238:0000:0000:0000:0000:0003:0001.- 14714 (32.078%)
2806:0000:0000:0100:0000:0000:0000:0019.- 18663 (40.687%)
Suma=45,870 total=45,870
Análisis de Path
Promedio Path= 3.845
Mínimo Path  = 1 (6503)
Máximo Path  = 29 2400:C700:3001::/48.- (22566 3491 6453 55644 55644 55644 55644 55644
55644 55644 55644 55644 55644 55644 55644 55644 55644 55644 55644 55644 55644 55644
55644 55644 55644 55644 55644 55644 55644)
Los Sistemas Autónomos inician con:
6503 (Axtel, S.A.B. de C.V., MX)
22566 (Maxcom Telecomunicaciones, S.A.B. de C.V., MX)
18734 (Operbes, S.A. de C.V., MX)
Buscando Errores por ISP
6503 (Axtel, S.A.B. de C.V., MX)
Rutas no optimizadas:
1/1/45870.-         2806:000F::/32
(2001:1278:0001:0000:0000:0000:0000:00f2) 18734 6939 6503
Rutas con AS Privados
2/2/45870.-         2001:067C:0470::/48
(2806:0000:0000:0100:0000:0000:0000:0019) 6503 174 8758 64513
3/3/45870.-         2001:067C:2E04::/48
(2806:0000:0000:0100:0000:0000:0000:0019) 6503 174 15945 65001
4/4/45870.-         2001:16A0::/29
(2806:0000:0000:0100:0000:0000:0000:0019) 6503 174 39386 263 25019 39891
64785 64787 64803 64905 65000
5/5/45870.-         2620:0000:1A50::/48
(2806:0000:0000:0100:0000:0000:0000:0019) 6503 174 32360 270 65010 65020
65030 65040 65050 65070 65090 65100 65130 65150 65160 65190 65230 65240
6/6/45870.-         2A02:2190::/32
(2806:0000:0000:0100:0000:0000:0000:0019) 6503 174 44134 65101
22566 (Maxcom Telecomunicaciones, S.A.B. de C.V., MX)
Rutas no optimizadas:
Rutas con AS Privados
18734 (Operbes, S.A. de C.V., MX)
Rutas no optimizadas:
Rutas con AS Privados
```

```
>quit
Quiting...
Quiting OK
```

B.1.3. Análisis de rutas IPv4 desde la Universidad Panamericana 2020

Este es el resultado de analizar rutas de IPv4 el 20/08/2020

Listado B.3: Resultados rutas UP IPv6 2020

```
Atendiendo lap-19643:179...
>
>connect 192.100.201.202
>Conexión IPv4
Version 4
BGP ID 192.100.201.202/32
Num. Parámetros 5
Recibiendo OPEN
Conectado a /192.168.5.224 desde 192.100.201.202:179
AS = 13679
La conexión no está duplicada; Aceptando...

>st
Mostrando 192.100.201.202/32 (/192.100.201.202->179)
192.100.201.202/32 up for 00 days, 00 hours, 00 mins, 21 seconds.
Updates: 142,445  Announces: 832,926  Witdrawls: 0  Routes: 832,889
>st
Mostrando 192.100.201.202/32 (/192.100.201.202->179)
192.100.201.202/32 up for 00 days, 00 hours, 00 mins, 23 seconds.
Updates: 142,445  Announces: 832,926  Witdrawls: 0  Routes: 832,889
>revisa
Conexiones de 192.100.201.202/32 (/192.100.201.202->179) son:
Análisis de Next-Hop
192.100.201.254/32.- 2 (0.000%)
192.100.201.202/32.- 2 (0.000%)
201.161.34.97/32.- 120949 (14.522%)
148.245.154.1/32.- 256315 (30.774%)
162.97.148.93/32.- 455625 (54.704%)
Suma=832,893 total=832,895
Análisis de Path
Se encontraron 111,305 rutas diferentes en los 832,895 segmentos de red

Promedio Path= 4.168
Mínimo Path  = 1 (6503)
Máximo Path  = 43 174.243.32.0/20.- (3549 3356 701 22394 6167 22394 22394 22394 22394 22394
    ↪ 22394 22394 22394 22394 22394 22394 22394 22394 22394 22394 22394 22394 22394 22394
    ↪ 22394 22394 22394 22394 22394 22394 22394 22394 22394 22394 22394 22394 22394 22394
    ↪ 22394 22394 22394 22394 22394)

Los Sistemas Autónomos inician con:
3549 (LVLT-3549, US)
6503 (Axtel, S.A.B. de C.V., MX)
22566 (Maxcom Telecomunicaciones, S.A.B. de C.V., MX)
Errores por Vecino 3549 (LVLT-3549, US)
1/1/832895.-        104.245.58.0/24        (201.161.34.97/32)   22566 32098 6762 3549 40627
2/2/832895.-        131.196.196.0/22       (201.161.34.97/32)   22566 32098 6762 3549 265919
3/3/832895.-        161.108.128.0/24       (201.161.34.97/32)   22566 3491 3356 3549 3955
4/4/832895.-        161.217.184.0/24       (148.245.154.1/32)   6503 174 701 22284 3549
5/5/832895.-        161.217.218.0/24       (148.245.154.1/32)   6503 174 701 22284 3549
6/6/832895.-        167.249.93.0/24        (201.161.34.97/32)   22566 32098 6762 3549 28283 264480
    ↪ 264480 264480
7/7/832895.-        172.111.119.0/24       (201.161.34.97/32)   22566 32098 6762 3356 3549 54113
8/8/832895.-        177.87.61.0/24         (148.245.154.1/32)   6503 3549 262596 262648
9/9/832895.-        187.1.128.0/22         (201.161.34.97/32)   22566 32098 6762 3549 28283 28299
10/10/832895.-      187.108.196.0/24       (201.161.34.97/32)   22566 32098 6762 3549 53107
11/11/832895.-      189.125.124.0/23       (201.161.34.97/32)   22566 32098 6762 3549
12/12/832895.-      189.125.128.0/24       (201.161.34.97/32)   22566 32098 6762 3549
13/13/832895.-      189.125.132.0/24       (201.161.34.97/32)   22566 32098 6762 3549
14/14/832895.-      189.38.84.0/22         (201.161.34.97/32)   22566 32098 6762 3549 28299
15/15/832895.-      189.38.88.0/22         (201.161.34.97/32)   22566 32098 6762 3549 28299
16/16/832895.-      191.102.0.0/20         (201.161.34.97/32)   22566 32098 52320 4230 3356 3549
    ↪ 262220
```

```
17/17/832895.-      192.33.232.0/24         (201.161.34.97/32)    22566 3491 3356 3549 3955
18/18/832895.-      199.76.1.0/24           (201.161.34.97/32)    22566 3491 3356 3549
19/19/832895.-      200.186.119.0/24        (201.161.34.97/32)    22566 32098 6762 3549
20/20/832895.-      200.186.47.0/24         (201.161.34.97/32)    22566 32098 6762 3549
21/21/832895.-      206.57.64.0/21          (201.161.34.97/32)    22566 3491 3356 3549
22/22/832895.-      208.48.201.0/24         (201.161.34.97/32)    22566 3491 3356 3549
23/23/832895.-      208.49.152.0/22         (201.161.34.97/32)    22566 3491 3356 3549
24/24/832895.-      208.49.152.0/24         (201.161.34.97/32)    22566 3491 3356 3549
25/25/832895.-      209.237.197.0/24        (201.161.34.97/32)    22566 32098 6762 3549 13414 13414
26/26/832895.-      64.192.0.0/24           (201.161.34.97/32)    22566 3491 3356 3549 33548
27/27/832895.-      64.192.1.0/24           (201.161.34.97/32)    22566 3491 3356 3549 33548
28/28/832895.-      64.192.2.0/24           (201.161.34.97/32)    22566 3491 3356 3549 33548
29/29/832895.-      64.192.3.0/24           (201.161.34.97/32)    22566 3491 3356 3549 33548
30/30/832895.-      64.192.4.0/24           (201.161.34.97/32)    22566 3491 3356 3549 33548
31/31/832895.-      64.192.5.0/24           (201.161.34.97/32)    22566 3491 3356 3549 33548
32/32/832895.-      8.243.40.0/24           (201.161.34.97/32)    22566 32098 6762 3549 10753

Errores por Vecino 22566 (Maxcom Telecomunicaciones, S.A.B. de C.V., MX)
1/33/832895.-       107.170.0.0/17          (148.245.154.1/32)    6503 22566
2/34/832895.-       138.185.224.0/24        (148.245.154.1/32)    6503 174 32098 22566 262944 262944
3/35/832895.-       138.185.225.0/24        (148.245.154.1/32)    6503 174 32098 22566 262944 262944
4/36/832895.-       138.185.226.0/24        (148.245.154.1/32)    6503 174 32098 22566 262944 262944
5/37/832895.-       170.100.0.0/16          (148.245.154.1/32)    6503 22566
6/38/832895.-       181.191.245.0/24        (162.97.148.93/32)    3549 3356 3491 22566 265524
7/39/832895.-       189.85.100.0/22         (148.245.154.1/32)    6503 1299 22566 270110
8/40/832895.-       200.77.236.0/24         (148.245.154.1/32)    6503 22566
9/41/832895.-       201.174.254.0/24        (148.245.154.1/32)    6503 22566
10/42/832895.-      201.182.20.0/24         (148.245.154.1/32)    6503 1299 22566 265524
11/43/832895.-      216.245.192.0/19        (148.245.154.1/32)    6503 22566
12/44/832895.-      45.167.158.0/24         (162.97.148.93/32)    3549 3356 3491 22566 263157
13/45/832895.-      45.167.159.0/24         (162.97.148.93/32)    3549 3356 3491 22566 263157
14/46/832895.-      45.174.44.0/24          (162.97.148.93/32)    3549 3356 3491 22566 28428
15/47/832895.-      45.185.244.0/24         (148.245.154.1/32)    6503 22566 265602
16/48/832895.-      45.188.108.0/24         (148.245.154.1/32)    6503 22566 265607
17/49/832895.-      45.188.109.0/24         (148.245.154.1/32)    6503 22566 265607
18/50/832895.-      45.188.110.0/24         (148.245.154.1/32)    6503 22566 265607
19/51/832895.-      45.188.111.0/24         (148.245.154.1/32)    6503 22566 265607
20/52/832895.-      45.189.252.0/22         (148.245.154.1/32)    6503 22566 265613 265613
21/53/832895.-      45.239.78.0/24          (162.97.148.93/32)    3549 3356 3491 22566 263157
22/54/832895.-      45.239.79.0/24          (162.97.148.93/32)    3549 3356 1299 22566 263157
23/55/832895.-      61.160.247.0/24         (148.245.154.1/32)    6503 22566
24/56/832895.-      65.126.125.0/24         (148.245.154.1/32)    6503 22566
25/57/832895.-      67.227.128.0/17         (148.245.154.1/32)    6503 22566
Rutas con AS Privados de 22566 (Maxcom Telecomunicaciones, S.A.B. de C.V., MX)
26/58/832893.-      118.185.160.0/21        (201.161.34.97/32)    22566 32098 9498 55410 24558 65001
27/59/832895.-      181.189.153.0/24        (201.161.34.97/32)    22566 32098 262206 65500
28/60/832896.-      202.61.32.0/20          (201.161.34.97/32)    22566 32098 6762 38193 58470 23966
    ↪ 23966 23966 45779 64547 64555 64557 64561 64590 64592 64601 64604 64616 64632 64638
    ↪ 64782 64847 64863 64881 64888 64900 64905 64921 64922 65270 136974
29/61/832896.-      7.188.0.0/16            (201.161.34.97/32)    22566 32098 64892 65515 65515 65509
    ↪ 55259
>
```

B.1.4. Análisis de rutas IPv6 desde la Universidad Panamericana 2020

Este es el resultado de analizar rutas de IPv6 el 20/08/2020

Listado B.4: Resultados rutas UP IPv6 2020

```
Atendiendo lap-19643:179...
>Conexión IPv6
Version 4
BGP ID 192.100.201.202/32
Num. Parámetros 5
Recibiendo OPEN
Conectado a /2801:f0:20:f80b:3c20:f62a:35fc:e2f6 desde 2801:f0:20:e80b:0:0:0:1:36477
AS = 13679
La conexión no está duplicada; Aceptando...
>st
Mostrando 192.100.201.202/32 (/2801:f0:20:e80b:0:0:0:1->36477)
192.100.201.202/32 up for 00 days, 00 hours, 02 mins, 45 seconds.
Updates: 32,382  Announces: 92,093  Witdrawls: 20  Routes: 91,663
>
>revisa
Conexiones de 192.100.201.202/32 (/2801:f0:20:e80b:0:0:0:1->36477) son:
Análisis de Next-Hop
2801:00f0:0020:e80b:0000:0000:0000:0001.- 1 (0.001%)
2801:00f0:0020:0880:0000:0000:0000:0001.- 1 (0.001%)
2806:0000:0000:0100:0000:0000:0000:0019.- 10486 (11.439%)
2001:1238:0000:0000:0000:0000:0003:0001.- 35980 (39.252%)
2001:0450:2002:0236:0000:0000:0000:009d.- 45197 (49.307%)
Suma=91,665 total=91,665
Análisis de Path
Se encontraron 25,566 rutas diferentes en los 91,665 segmentos de red

Promedio Path= 4.045
Mínimo Path  = 1 (6503)
Máximo Path  = 43 2600:100A:B1C0::/44.- (3549 3356 701 22394 6167 22394 22394 22394 22394
    ↪ 22394 22394 22394 22394 22394 22394 22394 22394 22394 22394 22394 22394 22394 22394
    ↪ 22394 22394 22394 22394 22394 22394 22394 22394 22394 22394 22394 22394 22394 22394
    ↪ 22394 22394 22394 22394 22394 22394)

Los Sistemas Autónomos inician con:
3549 (LVLT-3549, US)
6503 (Axtel, S.A.B. de C.V., MX)
22566 (Maxcom Telecomunicaciones, S.A.B. de C.V., MX)
Errores por Vecino 3549 (LVLT-3549, US)
1/1/91665.-         2001:0450:2060::/48    (2806:0:0:100::19)    6503 3549
2/2/91665.-         2001:0450::/32         (2001:1238::3:1)      22566 3491 3356 3549
3/3/91665.-         2001:13B0:00:/34       (2001:1238::3:1)      22566 3491 3356 3549
4/4/91665.-         2001:1900:2340::/44    (2001:1238::3:1)      22566 32098 6762 3549
5/5/91665.-         2804:22C0:40:/36       (2806:0:0:100::19)    6503 3549 3356 263903 264120
6/6/91665.-         2804:32F0::/32         (2806:0:0:100::19)    6503 3549 262476 265106
7/7/91665.-         2A07:AA00:0088::/48    (2001:1238::3:1)      22566 3491 3356 3549
Errores por Vecino 6503 (Axtel, S.A.B. de C.V., MX)
1/8/91665.-         2806:000F::/32         (2001:450:2002:236::9d) 3549 3356 6939 6503
Rutas con AS Privados de 22566 (Maxcom Telecomunicaciones, S.A.B. de C.V., MX)
1/9/91665.-         2001:067C:0714::/48    (2001:1238::3:1)      22566 32098 2603 224 64603
2/10/91665.-        2001:0DF0:E680::/48    (2001:1238::3:1)      22566 32098 6762 9498 55410
    ↪ 55410 10029 136369 64514
3/11/91665.-        2001:1270:2FFF::/48    (2001:1238::3:1)      22566 32098 22908 65001
4/12/91665.-        2401:4900:0EE0::/44    (2001:1238::3:1)      22566 32098 9498 45609 64962
    ↪ 134426
5/13/91665.-        2401:4900:0EF0::/44    (2001:1238::3:1)      22566 32098 9498 45609 64962
    ↪ 134426
6/14/91665.-        2402:4640:00:/33       (2001:1238::3:1)      22566 32098 4788 38044 65401
    ↪ 4200012340 23736
7/15/91665.-        2402:4640:80:/33       (2001:1238::3:1)      22566 32098 4788 38044 65401
    ↪ 4200012342 23736
>
```

B.2. Router Metrored características y análisis de sus rutas

El ruteador de Metrored es un Cisco 7609s (4 GB Ram, Ver IOS 12.2(33r)SER)

- Alto rendimiento, con hasta 720 Gbps en un solo chasis, o 40 Gbps de capacidad por ranura
- Una selección de factores de forma diseñados específicamente para una alta disponibilidad
- Diseño Cisco I-Flex: una cartera de adaptadores de puertos compartidos (SPA) y procesadores de interfaz SPA (SIP) que controlan experiencias de voz, video y datos
- Conjunto escalable y extensible de capacidades de hardware y software para habilitar servicios inteligentes de Carrier Ethernet
- Control integrado de admisión de video llamadas con una calidad visual innovadora de experiencia tanto para transmisión como para video a pedido (VoD)
- Intelligent Services Gateway, que proporciona reconocimiento de aplicaciones y suscriptores escalables con capacidades de identidad multidimensional y controles de políticas
- 250880K bytes of ATA System CompactFlash 0 (Read/Write)

Este equipo tiene configurado el AS13591 y tiene 96 vecinos entre los que destacan los AS de México mostrados en la Tabla B.1. Aquí se muestran los Sistemas Autónomos Mexicanos con los que tiene conectividad Metrored. Es importante mencionar, que las rutas a cualquier segmento de estos sistemas no se reenvíen hacia algún ISP en USA ya que la latencia se vería afectada por enlace internacionales, principalmente se utiliza el protocolo IPv4.

Sistema Autónomo en México	Cantidad de rutas
6503 (Axtel, S.A.B. de C.V., MX)	567
17072 (TOTAL PLAY TELECOMUNICACIONES SA DE CV, MX)	463
14178 (Megacable Comunicaciones de México, S.A. de C.V., MX)	113
11172 (Alestra, S. de R.L. de C.V., MX)	95*
22908 (Sixsigma Networks México, S.A. de C.V., MX)	26
18734 (Operbes, S.A. de C.V., MX)	17
13579 (INFOTEC Centro de Investigación e Innovación en TIC, MX)	16
16531 (TOPNET SA de CV, MX)	16
22566 (Maxcom Telecomunicaciones, S.A.B. de C.V., MX)	10
14249 (Internet Engine, S.A. de C.V., MX)	6
21603 (Universidad La Salle, AC, MX)	4
Total	1332

Tabla B.1: Sistemas Autónomos en México conectados a Metrored sobre IPv4 y cantidad de rutas anunciadas.

* Este Sistema Autónomo anuncia prefijos privados.

B.2.1. Router Metrored IPv4

Este es el resultado de analizar rutas sobre IPv4 el 29/11/2019

Listado B.5: Resultados rutas Metrored IPv4

```
Now listening on lap-19643:179...
>
>Sending. OPEN: version: 4 AS: 65534 holdtime: 240 BGPID: 3232237024(192.168.5.224/32)
Sent. OPEN: version: 4 AS: 65534 holdtime: 240 BGPID: 3232237024(192.168.5.224/32)
OPEN: Waiting for header
OPEN: Waiting for body
OPEN: received body
Versión 4
BGP ID 189.204.107.235/32
# Parámetros 5
1.- 010400010001 Multiprotocol Extensions for BGP-4 (1)
2.- 8000 Reservado Cisco-Router Refresh 128 (128)
3.- 0200 Route Refresh Capability for BGP-4 (2)
4.- 830100 Reservado Cisco-Multisession (131)
5.- 41040000fffe Support for 4-octet AS number capability (65)
received OPEN
Conect to /189.204.106.206 from /189.204.106.205:34176
AS = 65534
Connection not a dup; proceeding.
Keepalive sent
st
Mostrando 189.204.107.235/32
189.204.107.235/32 up for 00 days, 00 hours, 00 mins, 05 seconds.
Updates: 103187  Announces: 659496  Witdrawls: 0  Routes: 659496
>st
Mostrando 189.204.107.235/32
189.204.107.235/32 up for 00 days, 00 hours, 00 mins, 07 seconds.
Updates: 103220  Announces: 659539  Witdrawls: 42  Routes: 659459
>revisa
Conexiones de 189.204.107.235/32 is:
Análisis de Next-Hop
206.223.118.101/32.- 1 (0.000%)
189.204.89.232/32.- 2 (0.000%)
200.57.64.237/32.- 3 (0.000%)
206.223.118.216/32.- 4 (0.001%)
201.130.68.232/32.- 5 (0.001%)
189.204.57.224/32.- 6 (0.001%)
189.204.49.226/32.- 7 (0.001%)
206.223.118.69/32.- 8 (0.001%)
189.204.51.226/32.- 9 (0.001%)
189.204.91.224/32.- 10 (0.002%)
189.204.20.224/32.- 11 (0.002%)
189.204.103.224/32.- 12 (0.002%)
189.204.30.73/32.- 13 (0.002%)
206.223.118.81/32.- 14 (0.002%)
206.223.118.142/32.- 15 (0.002%)
206.223.118.153/32.- 18 (0.003%)
206.223.118.219/32.- 20 (0.003%)
206.223.118.192/32.- 21 (0.003%)
189.204.50.227/32.- 22 (0.003%)
206.223.118.221/32.- 24 (0.004%)
189.204.89.236/32.- 26 (0.004%)
206.223.118.76/32.- 27 (0.004%)
189.204.89.254/32.- 28 (0.004%)
201.148.128.251/32.- 29 (0.004%)
206.223.118.25/32.- 32 (0.005%)
206.223.118.107/32.- 34 (0.005%)
206.223.118.199/32.- 39 (0.006%)
206.223.118.177/32.- 41 (0.006%)
206.223.118.59/32.- 42 (0.006%)
189.204.114.224/32.- 43 (0.007%)
```

```
206.223.118.85/32.- 44 (0.007%)
206.223.118.75/32.- 47 (0.007%)
206.223.118.228/32.- 49 (0.007%)
206.223.118.231/32.- 56 (0.008%)
206.223.118.195/32.- 60 (0.009%)
189.204.114.225/32.- 71 (0.011%)
200.57.64.238/32.- 72 (0.011%)
206.223.118.119/32.- 76 (0.012%)
206.223.118.147/32.- 97 (0.015%)
206.223.118.202/32.- 121 (0.018%)
206.223.118.16/32.- 148 (0.022%)
206.223.118.159/32.- 186 (0.028%)
206.223.118.118/32.- 212 (0.032%)
206.223.118.194/32.- 256 (0.039%)
206.223.118.206/32.- 270 (0.041%)
206.223.118.164/32.- 419 (0.064%)
72.14.242.112/32.- 443 (0.067%)
17.1.138.65/32.- 792 (0.120%)
206.223.118.40/32.- 894 (0.136%)
206.223.118.24/32.- 966 (0.146%)
206.223.118.37/32.- 1101 (0.167%)
201.148.154.224/32.- 1372 (0.208%)
201.148.143.75/32.- 1495 (0.227%)
189.204.107.224/32.- 2222 (0.337%)
206.223.118.158/32.- 3330 (0.505%)
189.204.114.228/32.- 10856 (1.646%)
38.101.160.82/32.- 164656 (24.968%)
63.218.22.9/32.- 468132 (70.987%)
Suma=659,009 total=659,459
Análisis de Path
Promedio Path= 3.602
Mínimo Path  = 1 (54540)
Máximo Path  = 31 138.121.146.0/24.- (18734 12989 23456 23456 23456
23456 23456 23456 23456 23456 23456 23456 23456 23456 23456 23456 23456
23456 23456 23456 23456 23456 23456 23456 23456 52821 52821 52821
52821 52821 52821 52821 52821)
Los Sistemas Autónomos inician con:
36408 (CDNETWORKSUS-02 - CDNetworks Inc., US)
53780 (AS-APPRIVER - APPRIVER LLC, US)
54113 (FASTLY - Fastly, US)
28454 (Servicios DR de Mexico SA de CV, MX)
47254 (UDN-CORE, NL)
54119 (ELAUWIT-NET - Elauwit, LLC, US)
22908 (Sixsigma Networks Mexico, S.A. de C.V., MX)
46562 (TOTAL-SERVER-SOLUTIONS - Total Server Solutions L.L.C., US)
6503 (Axtel, S.A.B. de C.V., MX)
53828 (NITEL - NETWORK INNOVATIONS, INC., US)
19754 (FNL-33-19754 - The Fusion Network, LLC, US)
36351 (SOFTLAYER - SoftLayer Technologies Inc., US)
14860 (AS-SMARTCOM - SMARTCOM TELEPHONE, LLC, US)
49544 (I3DNET, NL)
29834 (USTREAM - USTREAM.TV INC, US)
40731 (LATIN-IP - Latin IP LLC, US)
3491 (BTN-ASN - PCCW Global, Inc., US)
29791 (VOXEL-DOT-NET - Voxel Dot Net, Inc., US)
36236 (NETACTUATE - NetActuate, Inc, US)
14907 (WIKIMEDIA - Wikimedia Foundation Inc., US)
18705 (RIMBLACKBERRY - BlackBerry Limited, CA)
53678 (VASTCOM - Vast.com, Inc, US)
13414 (TWITTER - Twitter Inc., US)
21523 (LEASENET - LeaseNet, US)
22616 (ZSCALER-SJC1 - ZSCALER, INC., US)
6495 (HSBC Mexico, S.A., Institucion de Banca Multiple, Grupo Financiero HSBC, MX)
46786 (IPTRANSIT - IP Transit Inc., US)
23148 (TERRENAP - MCI Communications Services, Inc. d/b/a Verizon Business, US)
29884 (EQUINIX-MA-DA1 - Equinix, Inc., US)
33353 (GAIKAI - Gaikai, Inc., US)
41095 (IPTP, NL)
33517 (DYNDNS - Dynamic Network Services, Inc., US)
```

```
11172 (Alestra, S. de R.L. de C.V., MX)
15133 (EDGECAST - MCI Communications Services, Inc. d/b/a Verizon Business, US)
6939 (HURRICANE - Hurricane Electric, Inc., US)
54540 (INCERO - Incero LLC, US)
11976 (FIDN - Fidelity Communication International Inc., US)
7908 (BT LATAM Venezuela, S.A., VE)
27471 (SYMN - Symantec Corporation, US)
23393 (ISPRIME - ISPrime, Inc., US)
32098 (TRANSTELCO-INC - Transtelco Inc, US)
14609 (EQUINIX-CORP-AS - Equinix, Inc., US)
36730 (TRANSFORMYX - Transformyx, Inc., US)
32934 (FACEBOOK - Facebook, Inc., US)
63383 (AIRWAVENETWORKSINC-US - Airwave Networks Incorporated, US)
16276 (OVH, FR)
23404 (RITTERNET - Ritter Communications, Inc., US)
36049 (RISE-TX-AS36049 - JAB Wireless, INC., US)
26077 (NEXTLINK-TEXAS - Nextlink Broadband, US)
13445 (13445 - Cisco Webex LLC, US)
714 (APPLE-ENGINEERING - Apple Inc., US)
2906 (AS-SSI - Netflix Streaming Services Inc., US)
18734 (Operbes, S.A. de C.V., MX)
6307 (AMERICAN-EXPRESS - American Express Company, US)
11666 (NEXICOM-CA - Nexicom Inc., CA)
20940 (AKAMAI-ASN1, US)
19551 (INCAPSULA - Incapsula Inc, US)
63399 (DIALPAD - Dialpad, Inc., US)
19151 (WVFIBER-1 - WV FIBER, US)
32592 (HT-HB32592 - HuntTel, US)
32590 (VALVE-CORPORATION - Valve Corporation, US)
22566 (Maxcom Telecomunicaciones, S.A.B. de C.V., MX)
286 (KPN, NL)
63949 (LINODE-AP Linode, LLC, US)
2635 (AUTOMATTIC - Automattic, Inc, US)
35986 (NO - Allegiance Communications, LLC, US)
27294 (RESERVETEL - Reserve Long Distance Company, Inc, US)
27298 (REACH4 - REACH4 Communications, US)
13335 (CLOUDFLARENET - Cloudflare, Inc., US)
11796 (AIRSTREAMCOMM-NET - Airstream Communications, LLC, US)
13579 (INFOTEC Centro de Investigacion e Innovacion en Tecnologias, MX)
16531 (TOPNET SA de CV, MX)
27704 (Tiendas Comercial Mexicana, S.A. de C.V., MX)
21603 (Universidad La Salle, AC, MX)
174 (COGENT-174 - Cogent Communications, US)
60725 (O3B-AS O3b Networks - Internet Service Provider operating, NL)
4826 (VOCUS-BACKBONE-AS Vocus Connect International Backbone, AU)
12177 (ETS-TELEPHONE-COMPANY - ETS TELEPHONE COMPANY, INC., US)
10310 (YAHOO-1 - Yahoo!, US)
15169 (GOOGLE - Google LLC, US)
15054 (NETXBB-TX - Northeast Texas Broadband, LLC, US)
46455 (NLI-ISP - Network Lubbock, Inc., US)
16265 (LEASEWEB-NETWORK LeaseWeb Network B.V., NL)
62642 (BIGLEAF - Bigleaf Networks, Inc., US)
17072 (TOTAL PLAY TELECOMUNICACIONES SA DE CV, MX)
62887 (WHITESKY-COMMUNICATIONS - WhiteSky Communications, LLC., US)
62646 (WIRESTAR - WireStar, Inc., US)
3073 (CITI7 - Citicorp, US)
32212 (HSB-EDGE - Sky Fiber Internet, US)
10848 (DELL-SERVICES - Dell Marketing USA, L.P., US)
5009 (EATEL - EATEL, US)
14249 (Internet Engine, S.A. de C.V., MX)
31800 (DALNET - DALnet, US)
Buscando Errores por ISP
16477 (ACNIELSEN-AS - ACNIELSEN, US)
Rutas no optimizadas:
1/1/659489.-        1.232.49.0/24          (38.101.160.82/32)    174 4766 16477
2/2/659489.-        12.107.122.0/23        (38.101.160.82/32)    174 7018 16477
3/3/659489.-        12.180.165.0/24        (63.218.22.9/32)      3491 7018 16477
4/4/659489.-        12.5.67.0/24           (63.218.22.9/32)      3491 209 16477 16477 16477 16477
    ↪ 16477
```

```
5/5/659489.-        121.162.175.0/24        (38.101.160.82/32)    174 4766 16477
6/6/659489.-        138.108.0.0/24          (38.101.160.82/32)    174 7018 16477
7/7/659489.-        138.108.100.0/24        (38.101.160.82/32)    174 2686 16477
8/8/659489.-        138.108.103.0/24        (38.101.160.82/32)    174 2686 16477
9/9/659489.-        138.108.104.0/23        (38.101.160.82/32)    174 2686 16477
10/10/659489.-      138.108.106.0/23        (38.101.160.82/32)    174 2686 16477
11/11/659489.-      138.108.110.0/24        (63.218.22.9/32)      3491 6939 3212 16477
12/12/659489.-      138.108.111.0/24        (38.101.160.82/32)    174 34772 16477
13/13/659489.-      138.108.113.0/24        (38.101.160.82/32)    174 31042 16477
14/14/659489.-      138.108.114.0/24        (63.218.22.9/32)      3491 3356 15994 16477
15/15/659489.-      138.108.124.0/24        (38.101.160.82/32)    174 37100 16477
16/16/659489.-      138.108.136.0/24        (63.218.22.9/32)      3491 3257 9534 16477
17/17/659489.-      138.108.137.0/24        (63.218.22.9/32)      3491 1299 4657 16477
18/18/659489.-      138.108.140.0/22        (63.218.22.9/32)      3491 2914 6660 16477
19/19/659489.-      138.108.141.0/24        (63.218.22.9/32)      3491 2914 6660 16477
20/20/659489.-      138.108.143.0/24        (63.218.22.9/32)      3491 2914 6660 16477
21/21/659489.-      138.108.144.0/24        (63.218.22.9/32)      3491 701 703 16477
22/22/659489.-      138.108.148.0/24        (63.218.22.9/32)      3491 9299 16477
23/23/659489.-      138.108.15.0/24         (63.218.22.9/32)      3491 6461 62 30023 16477
24/24/659489.-      138.108.162.0/24        (38.101.160.82/32)    174 4766 16477
25/25/659489.-      138.108.163.0/24        (38.101.160.82/32)    174 4766 16477
26/26/659489.-      138.108.2.0/24          (63.218.22.9/32)      3491 3356 3549 16477
27/27/659489.-      138.108.24.0/24         (63.218.22.9/32)      3491 3356 3549 16477
28/28/659489.-      138.108.25.0/24         (63.218.22.9/32)      3491 3356 3549 16477
29/29/659489.-      138.108.26.0/24         (63.218.22.9/32)      3491 3356 3549 16477
30/30/659489.-      138.108.27.0/24         (63.218.22.9/32)      3491 6461 62 30023 16477
31/31/659489.-      138.108.28.0/24         (63.218.22.9/32)      3491 3356 3549 16477
32/32/659489.-      138.108.29.0/24         (63.218.22.9/32)      3491 3356 3549 16477
33/33/659489.-      138.108.30.0/24         (63.218.22.9/32)      3491 3356 3549 16477
34/34/659489.-      138.108.31.0/24         (63.218.22.9/32)      3491 3356 3549 16477
35/35/659489.-      138.108.35.0/24         (38.101.160.82/32)    174 16477
36/36/659489.-      138.108.36.0/24         (63.218.22.9/32)      3491 3356 3549 16477
37/37/659489.-      138.108.38.0/24         (38.101.160.82/32)    174 52320 27717 16477
38/38/659489.-      138.108.39.0/24         (189.204.114.228/32) 18734 8151 16477 16477 16477 16477
    ↪ 16477 16477
39/39/659489.-      138.108.4.0/24          (63.218.22.9/32)      3491 3356 3549 16477
40/40/659489.-      138.108.42.0/24         (38.101.160.82/32)    174 16477
41/41/659489.-      138.108.47.0/24         (63.218.22.9/32)      3491 3356 3549 16477
42/42/659489.-      138.108.48.0/23         (63.218.22.9/32)      3491 5400 2856 16477
43/43/659489.-      138.108.50.0/24         (38.101.160.82/32)    174 16477
44/44/659489.-      138.108.53.0/24         (63.218.22.9/32)      3491 7018 16477
45/45/659489.-      138.108.55.0/24         (63.218.22.9/32)      3491 3356 16477
46/46/659489.-      138.108.6.0/24          (63.218.22.9/32)      3491 6461 62 30023 16477
47/47/659489.-      138.108.63.0/24         (63.218.22.9/32)      3491 7843 20001 16477
48/48/659489.-      138.108.7.0/24          (63.218.22.9/32)      3491 6461 62 30023 16477
49/49/659489.-      138.108.96.0/21         (38.101.160.82/32)    174 2686 16477
50/50/659489.-      138.108.97.0/24         (38.101.160.82/32)    174 2686 16477
51/51/659489.-      138.108.98.0/24         (38.101.160.82/32)    174 2686 16477
14178 (Megacable Comunicaciones de Mexico, S.A. de C.V., MX)
Rutas no optimizadas:
1/52/659489.-       132.255.124.0/22        (38.101.160.82/32)    174 14178
2/53/659489.-       144.36.27.0/24          (38.101.160.82/32)    174 14178 21433
3/54/659489.-       161.22.40.0/21          (38.101.160.82/32)    174 14178
4/55/659489.-       170.231.228.0/22        (38.101.160.82/32)    174 14178
32212 (HSB-EDGE - Sky Fiber Internet, US)
Rutas no optimizadas:
1/8449/659489.-     162.210.104.0/21        (38.101.160.82/32)    174 32212
2/8450/659489.-     162.210.104.0/24        (38.101.160.82/32)    174 32212
3/8451/659489.-     162.210.105.0/24        (38.101.160.82/32)    174 32212
4/8452/659489.-     162.210.106.0/24        (38.101.160.82/32)    174 32212
5/8453/659489.-     162.210.107.0/24        (38.101.160.82/32)    174 32212
6/8454/659489.-     162.210.108.0/24        (38.101.160.82/32)    174 32212
7/8455/659489.-     162.210.109.0/24        (38.101.160.82/32)    174 32212
8/8456/659489.-     162.210.110.0/24        (38.101.160.82/32)    174 32212
9/8457/659489.-     162.210.111.0/24        (38.101.160.82/32)    174 32212
>
>quit
Quiting...
Quiting OK
```

Apéndice C

Tipos de Redes

C.1. Tipos y características

Tipos de redes y sus características	
Redes Personales	▪ Estas redes se limitan a la conectividad generalmente de un solo individuo con sus diferentes dispositivos tales como los periféricos de computadoras, que se suelen conectar por medio de USB, Bluetooth u otras tecnologías alámbricas o inalámbricas [Gratton, 2013, Misic and Misic, 2008]. ▪ Los PDAs, Smartphones y Smart TVs han aprovechado las redes personales para comunicar los diferentes dispositivos y permitir el intercambio de contactos telefónicos, música y contenido multimedia entre sí. ▪ Las velocidades, por ejemplo, que puede alcanzar Bluetooth ver 4.0, se encuentra en los rangos de 24 a 32 Mbps [Proesch and Daskalaki-Proesch, 2015, p. 168].
Redes de área local	▪ El estándar más utilizado es el Ethernet y con menos frecuencia las redes como Token Ring [Wells et al., 1984, p. 1-42], FDDI [Mills, 1995, p. 4] y ATM [Hura and Singhal, 2001, p. 386]. ▪ Son redes de propiedad privada se encuentran en un solo edificio o en un campus de pocos kilómetros de longitud. Se utilizan ampliamente para conectar computadoras personales y estaciones de trabajo en oficinas de una empresa y de fábricas para compartir recursos. ▪ Las LANs tradicionales transmiten a una velocidad de 10 Mbps hasta los 40 Gbps y tienen un retardo bajo del orden de microsegundos o nanosegundos.
Redes WIFI	▪ La creciente demanda de conectividad y movilidad ha generado que las redes inalámbricas o WIFI, estén sustituyendo prácticamente a las redes LAN. El primer estándar sobre 802.11 se publicó en 1997, 6 años después que inició su creación y hasta la fecha se han realizado una variedad de modificaciones o enmiendas con la finalidad de incrementar su funcionalidad. ▪ Al ser un medio compartido, la velocidad en estos ambientes heterogéneos, rara vez sobrepasa los 70 Mbps, aunque las velocidades actuales alcanzan 3466 Mbps lo cual mejora por mucho la velocidad inicial de 11 Mbps [Perahia and Stacey, 2013, p. 182].

Red Metropolitana	▪ Surgidas en el 2008, cada día son menos utilizadas debido a que ya no tienen definido claramente sus límites geográficos. ▪ Las tecnologías como "Fiber to the Home" (FTTH) que permiten ofrecer televisión por cable, telefonía, generalmente voz sobre internet (VOIP por sus siglas en inglés) e Internet, en un esquema más parecido a la conexión de una LAN. ▪ Las velocidades de estas redes, son las de mayor crecimiento en los últimos años. Pasaron de ser enlaces de cable módems con 2 ó 3 Mbps a velocidades de 200 Mbps a 1,000 Mbps con los sistemas basados en Redes Ópticas Pasivas con Capacidad de Gigabit o GPON [Prat, 2008, p. 44].
Área amplia	▪ Generalmente abarcan países o continentes y permiten la conexión de las diferentes redes LAN entre sí. ▪ Están soportadas por líneas de transmisión y elementos de conmutación que permiten las conexiones de las diferentes redes lógicas [Tanenbaum, 2003, p. 19]. ▪ En la actualidad, las redes WAN forman parte principal del "backbone" de Internet, y el protocolo para el intercambio de rutas utilizado por todos los equipos es el BGP.

Tabla C.1: Características de las redes físicas según su alcance geográfico

C.2. Analogía de WIFI con SDN

En las definiciones de SDN, se suelen comparar los dispositivos con los controladores de las redes WIFI.

Las controladoras son dispositivos centralizados, que buscan optimizar cobertura, y calidad de servicio, variando parámetros como son los canales o frecuencias de cada antena, así como la potencia de transmisión de cada una.

Otro de los problemas que se tiene que abordar está relacionado con la superposición de bandas en las redes WIFI. Las dos bandas en las que opera la red WIFI, son 2.4 GHz y 5 GHz. En la banda de 2.4 existen en México once canales y tienen un ancho de banda de 20 MHz y se encuentran separados 5 MHz entre sí, lo que genera un traslape en canales adyacentes. Por ejemplo, el canal dos, con una frecuencia central de 2417 MHz ocupa desde la frecuencia 2407 hasta la 2427, por lo que se traslapa con el canal uno. En general hay traslapes de canales si éstos tienen una diferencia menor o igual a cinco unidades. Por lo anteriormente explicado, los canales recomendables son: 1, 6 y 11 como se muestra en la Figura C.1.

Lo explicado en el párrafo anterior, implica que, si se desea proporcionar servicio a muchos usuarios en un mismo lugar, se tenga que acomodar las antenas de manera que estos canales se encuentren distribuidos de forma que no estén próximas las mismas frecuencias [Nachtigall et al., 2008, Murray et al., 2007] lo que puede implicar reducir las potencias de las antenas y con esto limitar el alcance por supuesto. Adicionalmente, en estas redes existe una latencia debido a escaneo de los diferentes canales cuando un cliente WIFI se mueve entre dos puntos de acceso y es ocasionada por la superposición de los canales en la banda de 2.4 GHz y puede afectar el rendimiento en canales adyacentes.

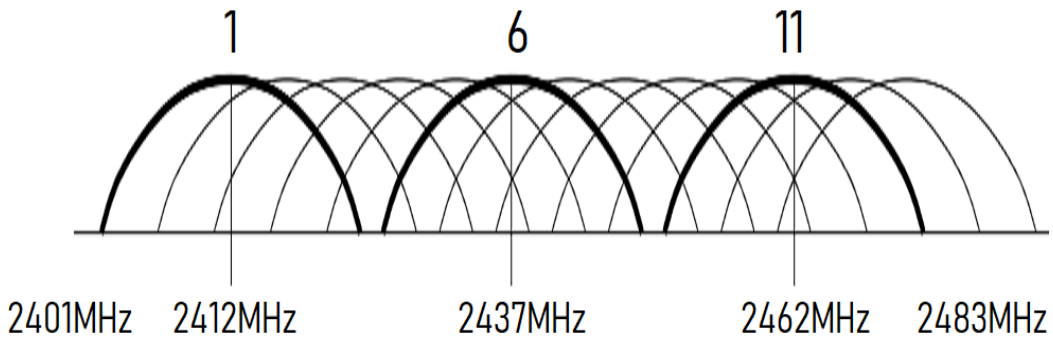

Figura C.1: Esquema de canales en las redes WIFI

[Fuxjager et al., 2007].

C.3. Protocolos de Enrutamiento

La Tabla, C.2 muestra un resumen de los protocolos, destacando sus aspectos más importantes, y si se usa para redes internas o externas [Mitchell, 2017]:

	Protocolos de Enrutamiento
Gateway Interior:	▪ RIP: Interno, se basa en la distancia más corta y utiliza el algoritmo Ford-Fulkerson. ▪ IGRP: Interno, más eficiente que RIP y se utiliza en redes muy grandes o complejas. ▪ OPFS: Interno, soporta enlaces punto a multipunto y utiliza el algoritmo Dijkstra. ▪ EIGRP: Interno, tiene el vector de distancias optimizado y utiliza el algoritmo Ford-Fulkerson. ▪ IS-IS: Interno, conexión de Sistema Intermedio a Sistema Intermedio y utiliza el algoritmo Dijkstra. ▪ iBGP: No es propiamente utilizado para redes Internas, pero si se usa en redes con el mismo AS. Es un subprotocolo de BGP basado en la distancia más corta y utiliza el algoritmo Dijkstra.
Gateway Exterior	▪ BGP: Estándar para todo Internet, utiliza el algoritmo Dijkstra y contempla trece parámetros para el cálculo de sus rutas, siendo el más importante la cantidad de saltos entre AS. ▪ eBGP: Externo y es un subprotocolo del BGP.

Tabla C.2: Clasificación de protocolos de ruteo.

Un ejemplo de cómo interactúan entre sí estos diferentes protocolos puede observarse en la Figura C.2, que muestra dos proveedores de Internet (ISP-1 e ISP-2) que conectan tres redes LAN (AS-1 a AS-3) cada una de ellas con su propio protocolo, todas unidas mediante BGP. El protocolo BGP, permite llegar a la frontera de cada una de las redes. En esta frontera, las rutas aprendidas por los protocolos de las redes LAN (OSPF, EIGRP y RIP), se encargan de distribuir los

paquetes internamente. Cuando un paquete dentro de alguna red Local, no conoce su destino, se envía al ruteado de frontera, quien al usar BGP, tiene la capacidad de enviarlo a cualquier dirección IP pública.

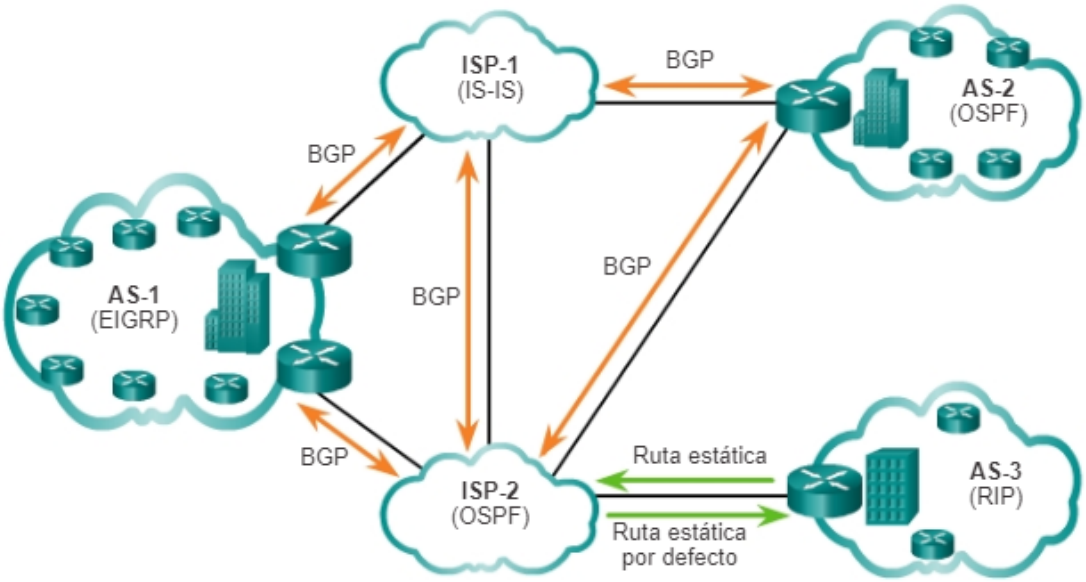

Figura C.2: Diagrama que muestra los alcances de los protocolos de enrutamiento.

[Cisco Networking Academy, 2014]

C.4. Herramientas de Redes

La principal tendencia en administración de redes es la de usar SDN, esto se debe a la versatilidad que proporciona para “Datacenters”, para implementar protocolos como el IPv6 [Jain et al., 2013, Martinez-Julia and F. Skarmeta, 2014] o el Internet de las Cosas. Empresas relacionadas con software y aplicaciones, ven en el SDN ventajas inmediatas y futuras, como es el caso de Citrix y Microsoft[Citrix, 2014, Schlesinger et al., 2015].

Actualmente existen soluciones como MIRO (Managed Internet Route Optimizer) [Internap, 2016], que proponen análisis distribuidos para evitar puntos de saturación, la evaluación de diferentes características de ruta para crear métricas de rendimiento que se utilizan para seleccionar las mejores rutas para los clientes de Internap (Internap es un proveedor global de centro de datos y soluciones en la nube). También actualmente, IPv6 (aunque tiene poco despliegue a nivel ISP), es usado por los proveedores de contenido, y el porcentaje de tráfico es muy importante, a mediados del 2019, ya llegaba al 62 % del tráfico total.

El proyecto Quagga, proyecto de código abierto, es una colección de protocolos que permiten implementar un router sobre el sistema operativo Linux. Quagga se puede usar perfectamente como herramienta de simulación. En el caso concreto de esta tesis, se utilizó mediante la creación de máquinas virtuales usando Oracle VM VirtualBox (Software de virtualización que permite instalar sistemas operativos adicionales en un mismo equipo), cada una de ellas implementan un router de BGP con su propio AS. Esto permitió en un solo equipo simular el router de la Universidad Panamericana (AS13679), sus tres proveedores de Internet, (AS6503, AS18734 y AS22566) y 2 o tres proveedores más conectados a los ISP de la UP.

Cada una de estas máquinas virtuales, puede funcionar perfectamente con memoria asignada de 1.5 a 4 GB de RAM, dependiendo del tipo de simulación y cantidad de segmentos de red a manejar y en el caso del almacenamiento, cada máquina es un clon enlazado de un equipo completo, por lo que prácticamente no requieren espacio adicional al equipo inicial que tiene un disco duro de 25 GB.

Cisco PT y VIRL son las herramientas de Cisco que van desde pequeños simuladores, creados para la academia de Cisco, como es el “Packet Tracer” (considerada la herramienta visual más difundida), hasta grandes desarrollos como VIRL (Cisco Virtual Internet Routing Lab), que permite simular en la nube de Cisco, grandes equipos y complejas configuraciones para simular grandes redes reales. VIRL es una potente plataforma de virtualización y orquestación de redes. Con él, los individuos y los equipos de TI pueden diseñar, construir, visualizar, solucionar problemas y ejecutar simulaciones de dispositivos de Cisco y de terceros en un entorno virtual [Janitor et al., 2010]. VIRL soporta los más modernos sistemas operativos de redes, de los que sobresalen: IOSv, IOSvL2, NX-OSv, NX-OS 9000v, IOS XRv, IOS

XRv 9000, IOS XE y ASAv.

Otra de las herramientas disponibles es “Linux Advanced Routing & Traffic Control”, basado en el sistema operativo Linux, que, con su experiencia en software libre, redes y presencia en ISP y grandes empresas ha adquirido madurez, eficiencia y confianza de los usuarios. Esta herramienta permite control de tráfico, generar rutas, procesos NAT, GRE (Túneles), IPsec, manejo de ancho de banda y latencia, es una excelente herramienta para la simulación de enlaces WAN, en donde se permite controlar el ancho de banda, la latencia y la pérdida de paquetes.

Apéndice D

Glosario

A

Access Point Equipo de red que se encarga de proporcionar la conectividad en las redes inalámbricas.

Actuadores Los actuadores son dispositivos capaces de transformar energía eléctrica, hidráulica o neumática con la finalidad de generar un efecto sobre un elemento externo. Cada día es más común, que estos dispositivos estén conectados en red y se controlen remotamente.

Algoritmo de Bellman-Ford Es un algoritmo que calcula las rutas más cortas desde un vértice de una sola fuente a todos los otros vértices en un grafo ponderado.

Algoritmo de Dijkstra Es un algoritmo para encontrar las rutas más cortas entre nodos de un grafo, creado por el científico informático Edsger W. Dijkstra.

Algoritmo de Ford-Fulkerson Es un algoritmo codicioso que calcula el flujo máximo en una red de flujo.

Algoritmos Genéticos Son algoritmos inspirados en el proceso de selección natural y pertenece a la clase más amplia de algoritmos evolutivos (EA). Los algoritmos genéticos se usan comúnmente para generar soluciones de alta calidad para la optimización y los problemas de búsqueda.

Algoritmos Híbridos Un algoritmo híbrido es uno que combina dos o más algoritmos que solucionan el mismo problema.

ALOHA Es un protocolo de acceso múltiple en la capa de enlace de datos y posibilita que múltiples terminales accedan al medio sin interferencia o colisión.

Ancho de Banda Es la tasa máxima de transferencia de datos a través de una ruta determinada.

Anova Herramienta estadística que se basa en el análisis de la varianza.

API Una interfaz de programa de aplicación (API por sus siglas en inglés) es un conjunto de rutinas, protocolos y herramientas para crear aplicaciones de software.

Archie Es una herramienta que indexa el contenido ofrecido por los distintos sitios FTP y le permite encontrar rápidamente un archivo determinado.

AS Abreviatura de "Autonomous System" es un identificador numérico que agrupa diferentes redes bajo un mismo esquema de administración.

ATM El modo de transferencia asíncrono (ATM por sus siglas en inglés) es un estándar para el transporte de una gama completa de tráfico de usuarios, incluidas señales de voz, datos y video. Fue desarrollado para satisfacer las necesidades de la red digital de servicios integrados de banda ancha, donde los paquetes son de tamaño fijo 53 bytes.

Azure Es una colección de servicios en la nube para crear, implementar y administrar aplicaciones inteligentes a través de una red global de centros de datos de Microsoft.

B

Balanceo de Cargas Mecanismo que mejora la distribución de las cargas de trabajo a través de múltiples recursos informáticos, como servidores, enlaces de red, unidades centrales de procesamiento o unidades de disco. Su principal objetivo optimizar el uso de los recursos, maximizar el rendimiento, minimizar el tiempo de respuesta y evitar la sobrecarga de cualquier recurso único.

BGP Border Gateway Protocol, es el protocolo predeterminado para difundir todas las redes en Internet.

Bluetooth Es un estándar de tecnología inalámbrica para intercambiar datos entre dispositivos fijos y móviles en distancias cortas utilizando ondas de radio UHF de onda corta en las bandas de radio de 2.400 GHz a 2.485 GHz.

C

Cache También conocido como "Memoria Cache", es un componente de hardware o software que almacena datos para que las futuras solicitudes de esos datos puedan atenderse más rápido.

CIDR "Classless Inter-Domain Routing" (Encaminamiento inter-dominios sin clases) es un método para asignar bloques de direcciones IP para remplazar al método que clasifica los segmentos en clases A, B y C.

Cloud Computing El cómputo en la nube, son esquemas que permiten ofrecer servicios de computación a través de una red, que usualmente es Internet.

Cluster Es un conjunto de computadoras conectadas de manera flexible o que funcionan juntas, de modo que, en muchos aspectos, pueden verse como un solo sistema.

Cómputo Paralelo Es un tipo de procesamiento en el que varios cálculos o la ejecución de varios procesos se llevan a cabo simultáneamente.

CPU Unidad Central de procesamiento, o "Central Processing Unit".

CSMA/CD Iniciales tomadas del nombre en inglés de "Carrier Sense Multiple Access with Collision Detection", es un algoritmo de acceso al medio compartido.

D

DMZ Iniciales tomadas del nombre en inglés de "Demilitarized Zone" es la zona intermedia entre las redes públicas de Internet y las redes privadas.

DTN Las redes tolerantes al retardo (DTN por sus siglas en inglés) es un enfoque de la arquitectura de red de computadoras que busca abordar los problemas técnicos en redes que pueden carecer de conectividad continua. Ejemplos de tales redes son aquellas que operan en entornos móviles o terrestres extremos, o redes planificadas en el espacio exterior.

DVR Iniciales tomadas del nombre en inglés de "Distance vector Routing", protocolo que requiere que los routers informen periódicamente a sus vecinos los cambios en la topología

E

eBGP BGP externo, el que se realiza con vecinos de sistemas autónomos diferentes.

Eficiencia Capacidad para realizar o cumplir adecuadamente una función, en esta tesis se utiliza como sinónimo de un buen diseño de redes.

EGP Iniciales tomadas del nombre en inglés de "Exterior Gateway Protocol",es el protocolo estándar para sistemas autónomos diferentes.

F

FDDI La interfaz de datos distribuida por fibra o “Fiber Distributed Data Interface”, es un conjunto de estándares sobre Fibra Óptica a 100 Mb/s con topología de anillo.

FTP Protocolo para la transferencia de Archivos.

G

Gopher Es un protocolo similar y contemporáneo al HTTP. Permite descargar archivos y conectarse a un servidor Telnet o un servidor de información.

Grafo Estructura matemática que, mediante una representación gráfica formada por nodos o vértices representa la relación entre objetos

GRE Es un protocolo que encapsula la información para su envío sobre túneles, del inglés: “Generic Routing Encapsulation”.

H

HTML Iniciales tomadas del nombre en inglés de“HyperText Markup Language” o Lenguaje de Marcas de Hipertexto, es el formato de los documentos de Hipertexto que se usan para HTTP.

HTTP El Protocolo de transferencia de hipertexto, es el protocolo más utilizado en Internet, es la base de todos los servicios de la WEB.

I

IANA Iniciales tomadas del nombre en inglés de “Internet Assigned Numbers Authority”, es la corporación que se encarga de la asignación global de direcciones IP y Sistemas Autónomos.

ICMP Iniciales tomadas del nombre en inglés de“Internet Control Message Protocol”, es el Protocolo de Mensajes de Control y Error de Internet, es un protocolo simple para la comunicación de error, marcas de tiempo o respuesta de accesibilidad

IOS Es el sistema operativo de Cisco, o “Cisco Internetwork Operating System”, proporciona servicios de conmutación en switches y de enrutamiento en routers.

Iot Iniciales tomadas del nombre en inglés de“Internet of Things”, es un concepto donde se agrupan dispositivos con conectividad a Internet, generalmente de forma inalámbrica y con la finalidad de automatizar su control.

IPSEC Es un conjunto de protocolos de red que autentica y encripta los paquetes de datos enviados a través de una red, esto incrementa la seguridad al evitar que un tercero, pueda ver la información que se envía.

IPv4 Protocolo de Internet Versión 4. Es el protocolo más utilizado en Internet, las direcciones son de 32 bits

IPv6 Protocolo de Internet Versión 6. Es el protocolo nuevo para Internet, las direcciones son de 128 bits

ISP Son organizaciones que brinda servicios para acceder, usar o participar en Internet. También conocidos como Proveedores de Internet.

ISR Iniciales tomadas del nombre en inglés de “Integrated Service Routers”, son equipos de redes para empresas pequeñas o con pocas redes.

L

LAN2LAN Es un tipo de redes virtuales sobre enlaces dedicados que hacen posible conectar dos sitios remotos como si se encontraran en la misma LAN.

Latencia En redes de datos, es el tiempo que transcurre desde que se genera el paquete en el origen hasta que se recibe en el destino.

LSR Iniciales tomadas del nombre en inglés de “Link State Routing” o Requerimientos de estado de enlace son paquetes de información que se intercambian los routers para actualizar sus tablas.

M

MED Iniciales tomadas del nombre en inglés de “Multi Exit Discriminator” de BGP, se utiliza en iBGP para influir en el destino cuando existe más de una salida de ese sistema autónomo.

MIMO La técnica de “Multiple Input Multiple Output” se utiliza en redes WIFI para aumentar el ancho del canal, utilizando técnicas de múltiples antenas o desfase de la señal.

MPLS Iniciales tomadas del nombre en inglés de “Multiprotocol Label Switching” o Conmutación Multiprotocolo Mediante Etiquetas es una técnica de enrutamiento en redes de telecomunicaciones que dirige los datos de un

nodo al siguiente, con base en etiquetas de ruta corta en lugar de direcciones de red largas, evitando así las búsquedas complejas en una tabla de enrutamiento.

MRTG Iniciales tomadas del nombre en inglés de "Multi Router Traffic Grapher" es una herramienta para monitorear variables SNMP, principalmente la carga de tráfico en los enlaces.

N

NAS Iniciales tomadas del nombre en inglés de "Network Attached Storage" es un tipo de almacenamiento que utiliza la misma infraestructura y los mismos protocolos que la red LAN.

NAT Iniciales tomadas del nombre en inglés de "Network Address Translation" es la traducción de direcciones de red es un método por el cual las direcciones IP son mapeadas de un dominio a otro, en un intento de proporcionar un enrutamiento transparente a los hosts. Tradicionalmente, los dispositivos NAT se utilizan para conectar un dominio de direcciones aislado con direcciones no registradas privadas a un dominio externo con direcciones registradas únicas a nivel mundial.

NTP Iniciales tomadas del nombre en inglés de "Network Time Protocol" es el protocolo para la sincronizar los relojes de las computadoras a través de una red.

O

OpenFlow Es un protocolo que permite a los controladores de red determinar la ruta de los paquetes de red a través de una red de conmutadores.

Optimizar Consiste en determinar los valores de las variables que intervienen en un proceso o sistema para que el resultado que se obtenga sea el mejor posible o se realice de la manera más eficiente.

OSI El modelo de interconexión de sistemas abiertos, también llamado OSI (Iniciales tomadas del nombre en inglés de "Open System Interconnection") es el modelo de red descriptivo propuesto por la Organización Internacional para la Estandarización compuesto por siete capas.

OSPF Iniciales tomadas del nombre en inglés de "Open Shortest Path First" es un protocolo de red para encaminamiento jerárquico de pasarela interior que usa el algoritmo Dijkstra.

P

POST En HTTP, el método POST se utiliza para enviar un elemento a un recurso en específico, se usa generalmente para crear un nuevo recurso.

PUT En HTTP, el método PUT se utiliza para enviar un elemento a un recurso en específico, se usa generalmente para modificar un recurso existente.

R

RFC Iniciales tomadas del nombre en inglés de "Request for Comments" son documentos donde se proponen estándares como protocolos, conceptos y metodología para Internet.

RIB Iniciales tomadas del nombre en inglés de "Routing Information Base" es la tabla con una selección de información de enrutamiento aprendida a través de protocolos de enrutamiento estático o de un protocolo de enrutamiento dinámico.

Roaming Es un término de redes inalámbricas que se usa típicamente con dispositivos móviles. Se refiere al dispositivo que se utiliza fuera del alcance de su red doméstica y se conecta a otra red celular disponible.

Router Es un equipo de red que se encarga de conectar con diferentes redes, generalmente con base en una tabla de ruteo, determina por cuál de sus interfaces debe de enviar los paquetes de datos.

Rutas Dinámicas Son los protocolos que permiten a un router entender la red y comunicar las rutas entre routers vecinos, para tomar la decisión de la mejor ruta.

S

SAN Iniciales tomadas del nombre en inglés de "Storage Area Network" es un tipo de almacenamiento que utiliza una infraestructura y protocolos distintos de la red LAN.

SDN Iniciales tomadas del nombre en inglés de "Software-defined networking" es una arquitectura de redes programable y flexible.

SDS Iniciales tomadas del nombre en inglés de "Storage Defined Networks" son herramientas para el aprovisionamiento basado en políticas y la gestión del almacenamiento de datos independientemente del hardware subyacente. El

almacenamiento definido por software normalmente incluye una forma de virtualización de almacenamiento para separar el hardware de almacenamiento del software que lo administra.

SNMP Iniciales tomadas del nombre en inglés de "Simple Network Management Protocol" es un protocolo de la capa de aplicación que facilita el intercambio de información de administración entre dispositivos de red.

V

VPN Las redes privadas virtuales son mecanismos de encapsulado de la información para trasmitirlas en Internet con mayor seguridad.

W

WAN Redes de área amplia.

WEB Iniciales tomadas del nombre en inglés de "World Wide Web", o simplemente Web se conoce a las páginas o documentos electrónicos en Internet vía HTTP.

WIFI Iniciales tomadas del nombre en inglés de "Wireless fidelity" es el conjunto de protocolos que proporcionan la conectividad a la red inalámbrica.

WLAN Redes inalámbricas de área local.

X

XML Iniciales tomadas en inglés de "eXtensible Markup Language" es un meta-lenguaje que nos permite definir lenguajes de marcado, utilizado en redes para la configuración de los dispositivos o el intercambio de datos.

Índice alfabético

Abstract, 3, **véase también** Resumen

Access Point, 208

Actuadores, 12, 208

Algoritmo, 55

Algoritmo de Bellman-Ford, 208

Algoritmo de Dijkstra, 11, 208

Algoritmo de Ford-Fulkerson, 208

Algoritmos Genéticos, 24, 208

Algoritmos Híbridos, 24, 208

ALOHA, 45, 208

Ancho de Banda, 46, 66, 88, 97, 207, 208

Anova, 71, 72, 77, 79, 97, 208

Antecedentes, 11

Análisis de varianza, **véase también** Anova

API, 183, 208

Aplicación, 80

Aplicación en la Industria, 98

Archie, 5, 208

AS, 7, **véase también** Sistemas Autónomos, 39, 55, 59, 101, 190, 193, 195, 201, 208

ATM, 201, 208

Azure, 208

Backbone, 46

Balanceo de Cargas, 24, 209

Base de información de enrutamiento, 22, 31

BGP, 7, 11, 57, **véase también** Protocolo BGP, 209

BGP Externo, 38, **véase también** eBGP

BGP Interno, 38, **véase también** iBGP

Bluetooth, 6, 201, 209

Cache, 10, 13, 27, 209

Camino más Corto, 25, 32, 33, 58

Caminos, 32

Caminos BGP, 32, 34

Canales WIFI, 203

CIDR, 15, 209

Cloud Computing, 25, 209

Cluster, 26, 209

Comparativo de rutas, 46

Comparativo Rutas BGP, 57

Comprobación de Hipótesis, 96

Conclusiones, 96

Conmutación de Circuitos, 14

Conmutación de Paquetes, 14

Contexto, 5

CPU, 209

CSMA/CD, 209

Cómputo Paralelo, 12, 209

DMZ, 210

DTN, 210

DVR, 210

eBGP, 30, 38, 210

Eficiencia, 6, 11, 25, 33, 210

EGP, 15, 210

EIGRP, **véase también** Protocolo EIGRP

Enrutamiento, 14, 23, 29, 30, 202, 204

Experimentación y Resultados, 63

Experimento Factorial, 63

FDDI, 201, 211

FTP, 211

Gateway, 29, 30

Gopher, 5, 211

Grafo, 34, 211

GRE, 211

Historia y Clasificación de las redes, 11

HTML, 211

HTTP, 211

IANA, 211

iBGP, 30, 38

ICMP, 23, 46, 68, 70, 88, 211

IGP, 15

Internet, 5, 7, 9, 29, 30, 46, 90

Internet: Red de Redes, 13

Intervalos de confianza, 44, 48, 97

Introducción, 5

Inyección de Rutas, 25, 59, 80

IOS, 211

Iot, 211

IPSEC, 211

IPv4, 16, 211

IPv6, 17, 211

IS-IS, **véase también** Protocolo IS-IS

ISP, 30, 37, 55, 57, 87, 90, 96, 101, 207, 211

ISR, 211

Justificación, 1

Justification, 3, **véase también** Justificación

LAN, 26, 47

LAN2LAN, 212

Latencia, 11, 45, 46, 49, 64, 66, 71, 75, 90, 97, 207, 212

Latencia Total, 47

LOCAL_PREF, 57

LSR, 212

Límites de Confianza, 54, 73, 75, 78, 79, 97

MED, 212

Mejor Ruta, 55

Mensajes BGP, 41

Methodology, 4, **véase también** Metodología

Metodología, 2

MIMO, 212

MPLS, 212

MRTG, 212

Máquina de estados de BGP, 31

Métricas, 9, 34, 46

NAS, 213

NAT, 213

Next-Hop, **véase también** Próximo Salto

NTP, 11, 213

Objetivo General, 7

Objetivos, 7

Objetivos Específicos, 8, 54, 57

OpenDaylight, 25, 94, 100

OpenFlow, 213

OPFS, **véase también** Protocolo OPFS

Optimización, 1, 3, 9, 11, 14, 23, 24, 27, 44–46, 71, 96

Optimizar, 213

OSI, 213

OSPF, 11, 213

Papers Publicados, 98

Path, **véase también** Caminos BGP

Paths, 32

Plano de Control, 22

Plano de Datos, 23

POST, 214

Problema, 6

Problemática, 1

Propuesta de la Tesis, 1

Protocolo BGP, 29

Protocolos de Enrutamiento, 204

Protocolos de Redes, 16

PUT, 214

Quagga, 206

Redes Cercanas, 72

Redes Lejanas, 77

Reflector de Ruta, 39

Resultados, 2

Resultados obtenidos, 98

Results, 4, **véase también** Resultados

Resumen, 1

Resumen del Estado Actual de la optimización de Redes, 27

RFC, 5, 17, 29, 32, 41, 42, 100, 214

RIB, **véase también** Base de información de enrutamiento, 214

RIP, 7, **véase también** Protocolo RIP

Roaming, 214

Router, 6–8, 12, 15, 17, 21–23, 25, 31, 37, 64, 70, 206, 214

Routing Information Base, 31, **véase también** Base de información de enrutamiento

Rutas Dinámicas, 214

Salto entre Sistemas Autónomos, 34

SAN, 214

SDN, 1, 3, 8, 20, 214

SDS, 214

Siguiente Salto, 37, **véase también** Próximo Salto

Sistemas Autónomos, 9, 32, 46, 59, 81, 101, 189, 201

SNMP, 214

Soluciones actuales, 11

Soluciones de SDN, 26

Tabla de Enrutamiento, **véase también** Base de información de enrutamiento

Tabla de Rutas, **véase también** Base de información de enrutamiento

Tiempo de propagación, 47

Tiempo de Trasmisión, 47

Tipos de BGP, 37, 40

Varianza Máxima, 53, 73, 75, 76

Varianza Mínima, 53, 73, 75, 76

VPN, 215

WAN, 1, 11, 12, 20, 27, 44, 45, 63, 66, 67, 69, 99, 202, 215

WEB, 215

WIFI, 215

WLAN, 215

XML, 215