Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Laboratorio de Comunicación y Redes

Implementación de un Protocolo de Enrutamiento Avanzado de Vector de Distancia

Trabajo Especial de Grado
presentado ante la Ilustre
Universidad Central de Venezuela
por los Bachilleres:

Valentina Trujillo Di Biase
V-17348234
valentina.trujillo@gmail.com

Jesús Expósito Marquina
V-17589761
exposito.j@gmail.com

para optar por el título de Licenciado en Computación.

Tutor: Prof. Eric Gamess

Caracas, Enero 2010

Universidad Central de Venezuela

Facultad de Ciencias
Escuela de Computación
Laboratorio de Comunicación y Redes

ACTA DEL VEREDICTO

Quienes suscriben, Miembros del Jurado designados por el Consejo de Escuela de Computación, para examinar el Trabajo Especial de Grado, presentado por los Bachilleres Valentina Trujillo Di Biase. C.I.: 17.348.234 y Jesús Expósito Marquina C.I.: 17.589.761, con el título "**Implementación de un Protocolo de Enrutamiento Avanzado de Vector Distancia**", a los fines de cumplir con el requisito legal para optar al título de Licenciado en Computación, dejan constancia de lo siguiente:

Leído como fue dicho trabajo por cada uno de los Miembros del Jurado, se fijó el día 27 de enero de 2010, a las 2:00 PM, para que sus autores lo defiendan en forma pública, en Planta Alta III de la Escuela de Computación, mediante la exposición oral de su contenido, y luego de la cual respondieron satisfactoriamente a las preguntas que le fueron formuladas por el Jurado, todo ello conforme a lo dispuesto en la Ley de Universidades y demás normativas vigentes de la Universidad Central de Venezuela. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo.

En fe de lo cual se levanta la presente Acta, en Caracas a los 27 días del mes de enero del año dos mil diez, dejándose también constancia de que actuó como Coordinador del Jurado el Profesor Tutor Eric Gamess.

Prof. Eric Gamess
(Tutor)

Prof. Robinson Rivas
(Jurado Principal)

Prof. Rafael Angulo
(Jurado Principal)

AGRADECIMIENTOS

Dedico este trabajo y todos sus frutos a mi familia, especialmente a mis hermanos y padres, gracias por permitirme llegar a esta nueva etapa de mi vida, todo mi cariño para ustedes.

Agradezco a la familia Rutigliano Abdelnour (en especial a la Sra. Ma. Cristina Abdelnour) por todo su apoyo y por los momentos gratos compartidos, los quiero!.

Gracias a Jonathan (Mazo) por su excepcional paciencia y cariño, te amo.

A mi tutor, Eric Gamess, por su constante disposición e invaluables enseñanzas.

A la UCV y a aquellos profesores que en realidad ayudaron a mi formación como profesional.

A José Hernández (el Chino), por su aporte y ayuda para la realización del paper basado en este trabajo.

A Rufo y King por su compañía durante el desarrollo de muchos aspectos de este producto.

De forma general, quiero agradecer a todas aquellas personas que hicieron posible la existencia de Easy-EIGRP.

A todos, MIL GRACIAS!

Valentina Trujillo Di Biase

AGRADECIMIENTOS

Al analizar el pasado, el trayecto recorrido, las vivencias experimentadas y la finalización de un trabajo tan difícil y lleno de problemas como el desarrollo de un Trabajo Especial de Grado, es inevitable recordar el aporte y la participación de todas aquellas personas que de no haber estado presentes hubiese sido imposible de lograr. Por ello, es un verdadero placer expresar mis agradecimientos a todas esas personas.

Quisiera empezar agradeciendo a mi hermana Susana (para que no te molestes), a ti, por ser un personaje único e irremplazable en mi vida, que desarrollas continuamente mis niveles (por lo visto ilimitados) de paciencia Y cariño. Gracias por tu personalidad incomparable que me hace aprender cada día más y más. Te agradezco desde el fondo de mi barriga cada comida clave y oportuna para tomar fuerzas y poder culminar este trabajo y te agradezco desde el fondo de mi corazón cada gota de sinceridad y cariño hacia mí. ¡Gracias por ser como eres!.

A Suso y a María (papá y mamá), por escuchar, observar y opinar aunque no tuviesen la más mínima idea de los que se trataba este proyecto, pero sobre todo, gracias por no molestarme durante todas mis horas de desarrollo frente a la computadora, que se traduce como, un apoyo incondicional y por confiar plenamente en mí.

A ti Gaby, por tu inmensurable paciencia y cariño, gracias por tu amistad absoluta y madurez. Por tu impecable objetividad y por abrirme los ojos una y otra vez. Gracias por separarme del monitor y obligarme a descansar. Eres lo máximo, te adoro y te quiero en exceso. U Rock!.

Un agradecimiento especial para mi Tutor, el Profesor Eric Gamess, por su incondicional ayuda ante cada problema y obstáculo encontrado, por haber confiado en nosotros y apoyarnos en cada momento, por su singular positividad y palabras de aliento. Gracias a usted por exigirnos, por no conformarse y sobre todo por plantearnos metas de gran dificultad y saber que podíamos cumplirlas.

A mi compañera de Tesis, Tina, por tu constante dedicación y entrega en este proyecto, por tu esfuerzo y responsabilidad absoluta. Gracias por tu continua energía para trabajar en equipo (¡y vaya que equipo!) y por esa paciencia descomunal que posees. Por tu voluntad de seguir adelante y tu nivel de organización, en fin gracias por todo y por mucho más.

Finalmente quisiera agradecer a todos mis amigos y amigas, gracias a todos ustedes por estar siempre presentes, por compartir cada momento especial y sobre todo por cada minuto de ocio y esparcimiento brindado para mantener una buena salud mental a lo largo de toda esta ¡odisea! (entiéndase por ello playa, rumbas, domingos de softball y futbol sala, viajes, salidas, más playa, etc.). Y a todos aquellos que de alguna u otra manera contribuyeron con este gran trabajo.

Gracias… Totales!.

Jesús Expósito Marquina

RESUMEN

TÍTULO:
Implementación de un Protocolo de Enrutamiento Avanzado de Vector Distancia.

AUTORES:
Valentina Trujillo Di Biase.
Jesús Expósito Marquina.

TUTOR:
Prof. Eric Gamess

El presente Trabajo Especial de Grado consiste en el desarrollo de una aplicación denominada Easy-EIGRP para la enseñanza del protocolo EIGRP (Enhanced Interior Gateway Routing Protocol) patentado por Cisco Systems. La aplicación provee un conjunto de herramientas que permiten depurar paso a paso todos los procesos llevados a cabo durante el funcionamiento de EIGRP en distintos escenarios.

El trabajo se inicia con un estudio basado en ingeniería inversa, el cual consiste en el desarrollo de situaciones puntuales para el estudio del comportamiento del protocolo utilizando routers Cisco. Durante la implementación de la aplicación, se utilizó una metodología iterativa basada en las nuevas tendencias del modelo de desarrollo ágil que plantea la creación de cada una de las herramientas de la aplicación en una iteración del proceso que abarca las fases de Análisis, Diseño, Implementación y Pruebas.

Easy-EIGRP está diseñado para ser instalado en cualquier computador y convertir al mismo en un router creando la posibilidad de comunicarse con otros routers o computadores que cuenten con el protocolo.

Easy-EIGRP es una implementación parcial de EIGRP donde se encuentra el protocolo RTP (Reliable Transfer Protocol), el algoritmo DUAL (Diffussing Update Algorithm), el protocolo de descubrimiento y recuperación de vecinos, entre otros. Además, la aplicación incorpora una gran cantidad de funcionalidades para facilitar el uso de la herramienta y el entendimiento del protocolo. Algunas de estas funcionalidades son: posibilidad de configuración de interfaces físicas y virtuales, mapas parciales de la red, máquina de estados finitos, sniffer, bitácoras de sucesos con distintos niveles de depuración, etc.

Palabras Clave: EIGRP, RTP, DUAL, Cisco Systems, Protocolo de Enrutamiento, Java.

Tabla de Contenido

Índice de Figuras

Índice de Tablas

Introducción

Un protocolo de enrutamiento es un mecanismo que especifica la forma en la que los routers intercambian información para calcular la mejor ruta hacia una red destino. Generalmente, los routers poseen un conocimiento a priori de sus vecinos inmediatos o de los llamados nodos adyacentes; gracias a la información recibida de éstos, un router podrá inferir información parcial o total de la topología de la red.

Un protocolo de enrutamiento utiliza distintas métricas para determinar qué ruta utilizará para transmitir un paquete a través de una red. Las métricas se calculan en base a ciertas propiedades de la red, como por ejemplo:

- Número de dispositivos a lo largo de la ruta (*hop count*).
- Ancho de banda.
- Retardos.
- Carga.
- Fiabilidad.
- MTU (*Maximum Transfer Unit*).

Los protocolos de enrutamiento pueden clasificarse de forma general según su alcance en:

- Protocolo Interior: destinado a trabajar dentro de un dominio administrativo, conocido como IGP (*Interior Gateway Protocol*) [21].
- Protocolo Exterior: destinado a trabajar entre dominios administrativos; este tipo de protocolo es comúnmente llamado EGP (*Exterior Gateway Protocol*) [7].

Otra clasificación posible se hace basándose en el tipo de información que intercambian los routers y la forma en que calculan las tablas de enrutamiento. En este caso se habla de:

- Protocolos de enrutamiento de vector de distancia.
- Protocolos de enrutamiento de estado del enlace.

Los protocolos de vector de distancia utilizan el algoritmo de Bellman-Ford y requieren que cada router simplemente informe a sus vecinos sobre su tabla de enrutamiento. Para cada red propagada, los routers destino deben elegir un anuncio entre los recibidos, el cual represente el menor costo para luego añadirlo a su tabla de enrutamiento. Algunos ejemplos de protocolos de este tipo son: RIPv1 (*Routing Information Protocol version 1*) [1], RIPv2 (*Routing Information Protocol version 2*) [12], IGRP (*Interior Gateway Routing Protocol*) [16], entre otros.

Los protocolos de vector de distancia son simples y eficientes en redes pequeñas y requieren muy poca administración, sin embargo presentan algunas limitaciones como por ejemplo, poca escalabilidad, lenta convergencia y falta de prevención de ciclos de enrutamiento por lo que sufren de cuentas al infinito.

Los protocolos de estado del enlace requieren que cada router mantenga al menos un mapa parcial de la red, para ello utilizan el algoritmo de Dijkstra. Cuando un enlace cambia de estado (de activo a inactivo o viceversa), la red es inundada con una notificación denominada LSA (*Link State Advertisement*). Todos los routers notan el cambio y recalculan las rutas de forma adecuada. Este método es confiable, sencillo de depurar y consume menos ancho de banda que los protocolos de vector de distancia. No obstante, los protocolos de este tipo son más complejos de implementar y requieren de mayor cantidad de memoria. Algunos protocolos de estado del enlace son: OSPF (*Open Shortest Path First*) [19] y IS-IS (*Intermediate System to Intermediate System*) [25].

En 1992 surgió un protocolo denominado EIGRP (*Enhanced Interior Gateway Routing Protocol*) propiedad de Cisco Systems el cual es basado en IGRP (*Interior Gateway Protocol*). EIGRP es caracterizado por ser un protocolo avanzado de vector de distancia dado que es una fusión entre los protocolos de vector de distancia y de estado del enlace. Gracias a esta propiedad, EIGRP logra mejorar el manejo de ciclos de enrutamiento y la escalabilidad; además, minimiza los tiempos de convergencia de la red y ofrece soporte para máscaras de tamaños variables y para actualizaciones parciales, gracias a dichas propiedades, el protocolo es conocido como un protocolo extremadamente ventajoso y eficiente.

El objetivo del presente Trabajo Especial de Grado es la creación de una aplicación llamada Easy-EIGRP basada en módulos que brinden un conjunto de funcionalidades para la enseñanza del protocolo EIGRP desarrollado por Cisco Systems.

Con este fin, se ha estructurado el informe en 5 capítulos que explican los diferentes aspectos tomados en cuenta durante la creación de la aplicación. A continuación, se ofrece un breve resumen del contenido de cada uno de estos capítulos:

- Capítulo 1 (El Problema): Plantea los diferentes enfoques desde los que puede ser estudiado el problema junto al análisis de la solución; mostrando en detalle los objetivos y el alcance del presente Trabajo Especial de Grado.
- Capítulo 2 (Marco Teórico): Presenta las bases teóricas que fueron estudiadas y sobre las que se fundamenta la creación de Easy-EIGRP.
- Capítulo 3 (Enhanced Interior Gateway Protocol): Describe la metodología utilizada y otros aspectos relevantes a tomar en cuenta para la implementación de Easy-EIGRP.
- Capítulo 4 (Marco Aplicativo): Explica en detalle las diferentes etapas del proceso de implementación mediante la metodología utilizada.
- Capítulo 5 (Especificaciones Técnicas): Establece algunas recomendaciones así como características que deben ser tomadas en cuenta al momento de la instalación y utilización de Easy-EIGRP.
- Capítulo 6 (Conclusiones): Presenta las conclusiones y recomendaciones obtenidas a partir del Trabajo Especial de Grado realizado.

1. El Problema

1.1 Planteamiento del Problema

EIGRP (*Enhanced Interior Gateway Protocol*) posee numerosas ventajas sobre muchos protocolos de enrutamiento, entre ellas, los bajos tiempos de convergencia y la alta manejabilidad de ciclos en la red.

Sin embargo, EIGRP es propiedad de Cisco Systems, debido a esto, no existe implementación alguna de código abierto para este protocolo de enrutamiento. En vista a esto, se desea implementar el protocolo EIGRP tanto para plataformas Windows como Linux, creando una herramienta con una interfaz gráfica y poniendo a disposición de los usuarios el código fuente (en caso de que Cisco Systems lo autorice).

1.2 Justificación del Problema

Actualmente las redes son necesarias en la mayoría de los sistemas y plataformas, es por ello que la presencia de protocolos de enrutamiento en las mismas es tan importante. EIGRP es un protocolo que posee numerosas y significantes ventajas, las cuales hacen justificable su uso y estudio, sin embargo, el protocolo resulta propietario, por ello existe la necesidad de emplear equipos Cisco Systems para su uso. Esta solución plantea eliminar tal necesidad y de esta forma poder utilizar EIGRP como protocolo de enrutamiento en routers basados en PC.

Es importante resaltar que la aplicación debe contemplar los requerimientos de manejo y usabilidad que cualquier aplicación desarrollada en la actualidad debe proveer, de esta forma se asegura que se le brinda al usuario una aplicación de calidad y que a la vez facilita su uso en las diversas tareas para las que fue creada.

La finalidad de la aplicación es netamente didáctica, permitiendo a los usuarios utilizar, estudiar en profundidad y analizar los procesos involucrados en EIGRP. De ser permitido por Cisco Systems, el código fuente de esta aplicación será distribuido en forma gratuitita para que los usuarios lo puedan modificar y mejorar.

1.3 Objetivos

1.3.1 Objetivo General

Implementar un protocolo de código abierto basado en EIGRP, que sea multiplataforma y orientado a la docencia.

1.3.2 Objetivos Específicos

- Obtener un amplio conocimiento de los algoritmos y mecanismos que son propios de EIGRP.
- Diseñar una interfaz gráfica basada en los principios de usabilidad que facilite la configuración del protocolo.
- Implementar el protocolo de enrutamiento.

- Poner a disposición del usuario un depurador encargado de registrar los diferentes eventos.
- Realizar pruebas para verificar el correcto funcionamiento.
- Permitir la portabilidad entre las plataformas Windows y Linux.

1.4 Alcance

La aplicación creada para el Trabajo Especial de Grado tiene el siguiente alcance:

- Tanto el manejo de los protocolos IPX y AppleTalk como la redistribución de rutas hacia redes que manejen dichos protocolos no se encuentran en el alcance del trabajo desarrollado.
- La versión de IOS utilizada fue la 12.4 y la versión del protocolo implementado es la 1.2, por lo que solo funcionará de forma correcta con routers que provean dicha versión.
- La aplicación debe ser instalada y ejecutada como usuario administrador ya que la misma requiere de la ejecución de comandos privilegiados.
- El software trabaja sólo sobre redes IP.
- La herramienta trabaja sobre el protocolo IPv4.
- La aplicación está diseñada para ser utilizada tanto en ambientes Windows (XP preferiblemente) como Linux (preferiblemente Ubuntu).
- Finalmente, la solución está enfocada únicamente a la docencia.

2. Implementaciones Existentes de Protocolos de Enrutamiento

Actualmente existen varias implementaciones de herramientas de enrutamiento que manejan varios protocolos, sin embargo ninguna de éstas tiene soporte para EIGRP (*Enhanced Interior Gateway Routing Protocol*). Estas herramientas pretenden promover soluciones innovadoras de enrutamiento y seguridad. A continuación, se describen las más populares.

2.1 Zebra

GNU Zebra es un software gratuito que gestiona TCP/IP basado en protocolos de enrutamiento. Es liberado como parte del proyecto GNU, y se distribuye bajo la Licencia Pública General de GNU. Soporta el protocolo BGP-4 (*Border Gateway Protocol 4)* [27], así como RIP, RIPv2 y OSPFv2 [19]. Zebra es único en su diseño debido a su modularidad ya que maneja un proceso para cada protocolo.

El proyecto Zebra se inició en 1996. La idea de Zebra se originó luego de una reunión entre Kunihiro Ishiguro y Yoshinari Yoshikawa. Ambos decidieron combinar recursos para crear el primer software de motor de enrutamiento basado en la Licencia Pública General de GNU. En la actualidad, Zebra está a punto de concluir con la liberación de la versión 1.0[1].

2.1.1 Características de Zebra

- Modularidad: Debido a la naturaleza multiproceso del software de Zebra, es fácil de actualizar y mantener. Cada protocolo se puede actualizar por separado, dejando operativos a los demás protocolos y al router.
- Velocidad: El enrutamiento de paquetes se llevará a cabo a un ritmo más rápido que con un software tradicional. Zebra permite a los routers una transferencia de datos más rápida. La necesidad de transferir grandes cantidades de datos está aumentando rápidamente a medida que el Internet crece, Zebra satisface esa necesidad.
- Fiabilidad: En el caso de algún error de cualquiera de los módulos del software, el router puede permanecer en línea y los demás demonios van a seguir funcionando. El fracaso puede ser diagnosticado y corregido, sin tener el router desconectado.

Como se puede ver en la Tabla 2.1, actualmente Zebra soporta varios RFC:

[1] http://www.zebra.org/what.html

RFC	Nombre	Estado
2453	RIP y RIPv2	Soportado
2080	RIPng	Soportado
2328	OSPF	Soportado
2460, 2373, 2463, 2464	IPv6	Soportado
2236	IGMP e IGMPv2	Soportado
1812	Routers Requirements	Soportado
1771	BGP-4	Soportado
1157	SNMP	Soportará
2011, 2012, 2013	MIB2	Soportará
1493	Bridge MIB	Soportará
1643	Ethernet MIB	Soportará
1757	RMON (4 groups)	Soportará
1724	RIPv2 MIB	Soportará
1850	OSPF MIB	Soportará
1657	BGP-4 MIB	Soportará
2037	Entity MIB	Soportará
2096	IP Forwarding Table MIBs	Soportará

Tabla 2.1: RFCs soportados por Zebra.

2.1.2 Especificaciones de Zebra

Actualmente están desarrollando Zebra en las siguientes plataformas:

- GNU / Linux 2.2.X y 2.4.X.
- FreeBSD 4.X.
- FreeBSD 5.X.
- NetBSD 1.6.X.
- OpenBSD 3.X.

Zebra trabaja con IPv6 para dar soporte a:

- FreeBSD.
- NetBSD.
- OpenBSD.
- GNU / Linux.

Entre los protocolos de enrutamiento soportados están los de la Tabla 2.2

Demonios Soportados	Descripción
bgpd	BGP-4 y BGP-4+
ripd	RIPv1, RIPv2
ripngd	RIPng
ospfd	OSPFv2
ospf6d	OSPFv3
zebra	Para la tabla de enrutamiento del kernel y la actualización de información de enrutamiento por encima de la redistribución entre los protocolos.

Tabla 2.2: Protocolos de enrutamiento soportados por Zebra.

2.2 Quagga

Quagga es una suite de software de enrutamiento que nace de GNU Zebra, proporcionando implementaciones de OSPFv2, OSPFv3, RIPv1, RIPv2, RIPng, BGP-4 e IS-IS para plataformas Unix[2].

Un sistema con Quagga instalado actúa como un router dedicado. Con Quagga, el computador intercambiará información de enrutamiento con otros routers utilizando protocolos de enrutamiento, dicha información es utilizada para actualizar la tabla de enrutamiento del kernel. Adicionalmente se puede cambiar dinámicamente la configuración y se puede ver la información de la tabla de enrutamiento desde un terminal.

Si se tiene una pequeña red o conexión xDSL, la configuración del software de enrutamiento Quagga es muy fácil, lo único que se tiene que hacer es configurar las interfaces y poner sólo un par de comandos sobre rutas y/o rutas por defecto. Si la red es bastante grande, o si la estructura de la red cambia con frecuencia, se pueden activar los protocolos de enrutamiento dinámicos (RIP, OSPF y BGP).

Actualmente, Quagga soporta protocolos de enrutamiento unicast comunes, sin embargo, protocolos de enrutamiento multicast, como BGMP, PIM-SM, PIM-DM podrán apoyarse en Quagga 2.0. En el futuro, el control de filtrado TCP/IP, el control de QoS y la configuración de DiffServ se añadirá a Quagga.

Los softwares de enrutamiento tradicional se basan en un proceso ya existente que proporciona todas las funciones de los protocolos de enrutamiento. Quagga toma un enfoque diferente, ya que está hecho a partir de una colección de varios demonios que trabajan juntos para construir la tabla de enrutamiento. Adicionalmente puede haber varios protocolos de enrutamiento y el gestor de enrutamiento del kernel de Zebra.

De este modo el demonio ripd maneja el protocolo RIP, mientras que ospfd es un demonio que soporta OSPFv2, adicionalmente bgpd apoya el protocolo BGP-4. Para cambiar la tabla de enrutamiento del kernel y para la redistribución de rutas entre los diferentes protocolos de enrutamiento, hay un demonio gestor de la tabla de enrutamiento del kernel Zebra, como lo muestra la Figura 2.1. Es fácil añadir un nuevo demonio que maneje un nuevo protocolo de enrutamiento sin afectar a ningún otro demonio. Cabe destacar que se necesita ejecutar sólo el demonio del protocolo relacionado con los protocolos de enrutamiento en uso. Además, el usuario puede ejecutar un demonio de enrutamiento y enviar informes a una consola central de enrutamiento.

No hay necesidad de que estos demonios se ejecuten en el mismo computador, se puede incluso ejecutar varios demonios iguales en el mismo computador, por ello, esta arquitectura crea nuevas posibilidades para el sistema de enrutamiento.

[2] http://www.quagga.net/about.php

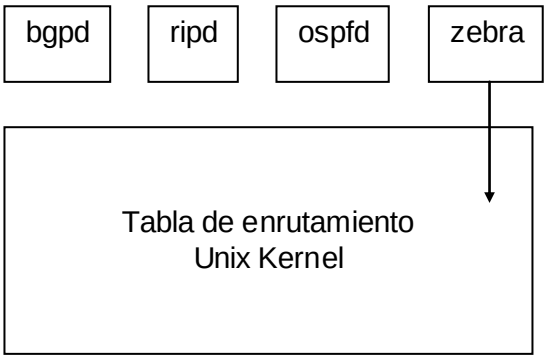

Figura 2.1: Arquitectura de Quagga.

Las plataformas soportadas son:

- GNU/Linux 2.4.x y superior.
- FreeBSD 4.x y superior.
- NetBSD 1. y superior.
- OpenBSD 2. y superior.
- Solaris 8 y superior.

2.3 XORP

XORP (*Extensible Open Router Platform*) es una suite de software de enrutamiento de código abierto, con el objetivo de ser estable para su uso en producción y también extensible para apoyar la investigación de redes. El proyecto fue fundado por Mark Handley en el año 2000, con la primera liberación en julio de 2004. El proyecto está ahora encabezado por Atanu Ghosh del International Computer Science Institute, en Berkeley, California[3].

A partir del 2007, el código da soporte a los siguientes protocolos de enrutamiento:

- BGP, incluyendo las extensiones multiprotocolo para IPv6.
- RIPv2 para IPv4 e RIPng para IPv6.
- OSPFv2 (RFC 2328) y OSPFv3 (RFC 2740) tanto para IPv4 como IPv6.
- PIM-SM (*Protocol Independent Multicast – Sparse Mode*) tanto para IPv4 como para IPv6.
- IGMPv1 (*Internet Group Management Protocol*), IGMPv2 y IGMPv3 (sólo para IPv4).
- MLDv1 (*Multicast Listener Discovery*) y MLDv2 (sólo para IPv6).

El código de XORP consta de alrededor de 670.000 líneas escritas en C++. XORP es soportado en plataformas FreeBSD, Linux, OpenBSD, DragonFlyBSD, NetBSD, MacOS X, incluso existe una adaptación para Windows Server 2003, que sólo es compatible con IPv4.

[3] http://www.xorp.org

2.4 Vyatta

Vyatta es una solución de routers, firewall y VPN (Virtual Private Network) que pueden ayudar a construir de forma asequible redes de clase empresarial ofreciendo el rendimiento, la fiabilidad y las características de los dispositivos de redes propietarios[4].

Debido a que uno de los mayores problemas en cuanto a redes y equipos es el costo, la posibilidad de tener un router con muy buenas características y a un buen precio es importante y posible con Vyatta.

Vyatta dispone de una versión gratuita (denominada *Community Edition*) que actualmente se encuentra en la versión 4.1.4. También existen versiones comerciales (*Professional, Enterprise* y *Premium*) que incluyen soporte y actualizaciones.

Este router soporta interfaces Ethernet 10/100/1000 e interfaces T1/E1 (para enlaces WAN). También soporta los protocolos de enrutamiento más comunes (OSPF, BGP-4, RIPv2 y rutas estáticas) e incluye un firewall y software para VPNs y NAT (*Network Translation Protocol*). Vyatta es administrable vía SNMP (*Simple Network Management Protocol*) y configurable vía línea de comandos o vía HTTP (*HyperText Transfer Protocol*). Adicionalmente Vyatta permite analizar el tráfico que transita por él mediante Wireshark.

La Figura 2.2 muestra las especificaciones técnicas de este software, incluyendo información acerca de seguridad, calidad de servicio, administración, *debugging*, entre otras características adicionales.

2.5 Click Modular Router

Click es un software de construcción y configuración de routers que implementa funciones de clasificación de paquetes, colas, *scheduling* e interfaces de dispositivos de red. Click es totalmente *open-source* y está abierto a cualquier tipo de contribución por parte de terceros[5].

Un router Click se ensambla de paquetes de módulos de procesamiento llamados elementos. Estos elementos controlan cada aspecto del comportamiento del router, desde la comunicación con los dispositivos hasta la modificación de paquetes, colas, etc. Estos elementos individuales pueden llegar a tener un sorprendente y poderoso comportamiento, adicionalmente, es muy fácil escribir nuevos elementos en C++. Básicamente la idea es configurar el router añadiendo elementos con un lenguaje simple.

[4] http://www.vyatta.com
[5] http://read.cs.ucla.edu/click/click?do=show

ESPECIFICACIONES TÉCNICAS

Soporte de Hardware
- 32-bit x86 processors
- PC architecture

Interfaces
- 10/100/1000 Ethernet cards
- 10 GbE cards
- T1/E1 cards – 1 port
- T1/E1 cards – 2 port
- T1/E1 cards – 4 port
- T3 cards
- X.21 and V.35*

  See Vyatta Hardware Catalog for all tested and supported interfaces

Protocolos de Enrutamientos e IP
- IPv4
- OSPFv2
- BGPv4
- RIPv2
- Static routes

Manejo de Direcciones IP
- Static
- DHCP server
- DHCP client
- DHCP relay

Encapsulamientos
- Ethernet
- 802.1Q VLANs
- PPP
- PPPoE
- MLPPP
- Frame Relay
- HDLC
- GRE, IP-in-IP

Balanceo de Carga
- WAN link load balancing
- MLPPP
- ECMP

Qos
- Priority Queuing
- Classful Queuing
- Bandwidth Management

Seguridad
- Stateful inspection firewall
- Network address translation
- IPSec VPN
- Remote VPN (PPTP, L2TP, IPSec)
- DES, 3DES, AES Encryption
- MD5 and SHA-1 Authentication
- RSA, Diffie Helman Key Management
- NAT Transversal
- RADIUS authentication
- Role based access control
- Intrusion Prevention*
- Anti-Virus*

Alta Disponibilidad de
- VRRP (Virtual Router Redundancy Protocol)
- IPSec VPN Clustering
- Protocol fault isolation

Administración
- Integrated CLI
- Web GUI**
- Single configuration file
- Telnet
- SSHv2, Network upgrades

Depuración y Sniffing
- Tcpdump
- Wireshark packet capture
- BGP MD5 Support

Logging y Monitoreo
- Syslog
- SNMPv2c

* Características experimentales, actualmente no soportadas

** Web GUI Q4 ‘08

Figura 2.2: Especificaciones técnicas de Vyatta.

Click fue originalmente desarrollado en el MIT (*Massachusettes Institute of Technology*), con un desarrollo posterior en Mazu Networks, ICIR (*Internet Communications and Information Research*) y UCLA (*University of California at Los Angeles*). Actualmente se encuentra en la versión 1.8.0 (liberada el 28 de febrero del 2010).

La distribución de Click incluye más de 300 elementos, el módulo del kernel de Linux, el *driver user-level*¸ el módulo del kernel FreeBSD, herramientas y documentación, todo bajo la licencia MIT/BSD-like. El núcleo de la distribución contiene los *drivers* de Click y una larga colección de elementos.

2.6 BIRD Internet Routing Daemon

El proyecto BIRD[6] tiene como objetivo desarrollar un completo y funcional demonio de enrutamiento dirigido principalmente a los sistemas tipo Unix. BIRD se distribuye bajo la licencia GNU *General Public License*.

BIRD fue desarrollado como un proyecto universitario en la Facultad de Matemáticas y Física, en la Universidad Charles de Praga. Actualmente es desarrollado y respaldado por los laboratorios de CZ. NIC Labs.

Los módulos que conforman BIRD están enlazados estáticamente para producir un único archivo ejecutable.

El proceso de creación es controlado por un conjunto de *Makefiles* para GNU *Make*, mezclados con varios *shell* y *Perl scripts*.

La configuración inicial del demonio, la detección de las características del sistema y la selección de los módulos correctos para un sistema operativo en particular y del conjunto de protocolos que el usuario haya elegido es realizada por un *script* de configuración generado por GNU *autoconf*.

BIRD soporta actualmente:

- IPv4 y IPv6.
- Múltiples tablas de enrutamiento.
- BGP.
- RIP.
- OSPF.
- Rutas Estáticas.
- Interfaz de línea de comandos.

[6] http://bird.network.cz/?index

3. EIGRP (Enhanced Interior Gateway Routing Protocol)

EIGRP, como su nombre lo indica, es una mejora del protocolo IGRP (*Interior Gateway Routing Protocol*) que al igual que EIGRP es propiedad de Cisco Systems. EIGRP continúa siendo un protocolo de tipo vector de distancia y utiliza la misma composición de métricas que IGRP [16].

Distintas propiedades se encuentran asociadas a una ruta o a un *vector metric* EIGRP:

- Ancho de banda mínimo en la ruta hasta la *subnet* destino.
- Retardo total desde el router hasta la *subnet* destino.
- Carga mayor del enlace en la ruta hasta la *subnet* destino.
- Fiabilidad mínima del enlace en la ruta hasta la *subnet* destino.
- MTU (*Maximum Transfer Unit*) mínimo del enlace en la ruta hasta la *subnet* destino.
- *Hop count*.

El *vector metric* es utilizado en conjunto con los valores K, para calcular un único número llamado *composite metric*, conocido también como distancia. Este número es utilizado en todas las comparaciones que se llevan a cabo cuando un router está tratando de decidir una mejor ruta.

El *vector metric* es entonces un vector compuesto de seis elementos o parámetros (ancho de banda, retardo, carga, fiabilidad, hop count y MTU) que describe la distancia entre un router y la *subnet* destino. El *vector metric* es utilizado en todos los *updates* de rutas EIGRP.

El c*omposite metric* o distancia es un número entero utilizado para comparar distintas rutas con un destino común. Sólo es utilizado internamente en los routers y nunca es enviado a vecinos EIGRP.

Los valores K son números (de K_1 a K_5) utilizados en la transformación de *vector metric* a *composite metric*. La fórmula para transformar un *vector metric* en un *composite metric* es la siguiente (tomada de [16]):

$$[1]\ \textit{Composite Metric} = 256 * ((K_1 * \text{Bandwidth}) + (K_2 * \text{Bandwidth})/(256 - \text{Load}) + (K_3 * \text{Delay}))$$

Adicionalmente, al *composite metric* debe ser multiplicado por:

$$[1]\ 256 * K_5/(\text{Realibility} + K_4)$$

siempre y cuando K_5 sea distinto de cero.

Donde el *bandwidth* es igual a 10 Gbps/ancho de banda (debe ser medido en bits por segundo) y el retardo debe ser expresado en decenas de microsegundos.

Debido a que los valores por defecto de las constantes K_i son los siguientes:

$$K_1 = 1$$
$$K_2 = 0$$
$$K_3 = 1$$
$$K_4 = 0$$
$$K_5 = 0$$

La métrica compuesta de EIGRP derivada es (tomada de [4]):

$$[2] \text{ Métrica} = [\min(\text{Bandwidth}) + \sum \text{Delay}\,] * 256.$$

En general, K_2, K_4 y K_5 no son utilizados. Esto se debe a que estos valores no funcionan correctamente con EIGRP. IGRP utiliza *updates* de rutas periódicos que reflejan las condiciones de carga y de fiabilidad del enlace, mientras que EIGRP utiliza *updates* sólo cuando ciertas acciones específicas son llevadas a cabo. Por ello, la carga y la fiabilidad en el *vector metric* de EIGRP son inútiles y no tiene sentido tomarlas en cuenta en la *composite metric*.

Muchas veces EIGRP es descrito como un protocolo de vector de distancia que actúa como un protocolo de estado del enlace. Como es sabido, los protocolos de vector de distancia comparten toda la información que manejan pero sólo con sus vecinos directos; en cambio, los protocolos de estado del enlace anuncian a toda la red únicamente información sobre sus nodos adyacentes [20].

A pesar de las diferencias que ambos tipos de protocolos presentan, poseen un común denominador: ambos calculan rutas de forma individual; con la diferencia que EIGRP hace uso de *diffusing computations* (cómputos distribuidos), es decir, los cálculos de las rutas son hechos en forma conjunta y coordinada entre múltiples routers. Este tipo de cálculo permite que la red converja rápidamente mientras que se mantiene libre de ciclos.

Aunque los mensajes de actualización de EIGRP continúan siendo vectores de distancias que son transmitidos a sus vecinos directamente conectados, estos se caracterizan por ser parciales, delimitados y no periódicos. La parcialidad de las actualizaciones se refiere a que sólo se incluirán las rutas que han cambiado y no todas las entradas de la tabla de enrutamiento, como se hace tradicionalmente. El hecho de que no sean periódicos quiere decir que las actualizaciones no son enviadas entre intervalos de tiempo regulares, únicamente son enviadas cuando se detecta un cambio de métrica o de topología. La delimitación significa que las actualizaciones sólo se envían a los routers afectados.

Las características antes mencionadas conllevan a que EIGRP consuma mucho menos ancho de banda que los protocolos de vector de distancia.

En líneas generales el tráfico en una red se hace muy alto durante períodos de convergencia, sin embargo, EIGRP no utiliza más del 50% del ancho de banda de un enlace en ningún momento (propiedad que puede ser configurada de forma manual).

Cada vez que EIGRP encola un paquete para ser transmitido por una interfaz, éste utiliza la fórmula siguiente (tomada de [8]) para determinar cuánto tiempo debe esperarse antes de enviar el paquete:

[3] ((1/anchoDeBanda) x (tamañoDelPaqueteEnBytes x 8))
x %delAnchoDeBandaPermitido

EIGRP es un protocolo de tipo *classless*, lo que quiere decir que las máscaras de *subnet* son enviadas en las actualizaciones. Las máscaras de tamaños variables pueden ser utilizadas con EIGRP no sólo para *subnetting* sino para el proceso de *summarization* (ver Sección 3.1.8). Los paquetes EIGRP pueden ser autenticados utilizando *checksums* criptográficos MD5 (ver Sección 3.2). Finalmente, una de las características de mayor relevancia de EIGRP es que no sólo puede enrutar sobre IP sino también sobre IPX y AppleTalk.

3.1 Funcionamiento de EIGRP

EIGRP utiliza la misma fórmula que IGRP para el cálculo de las métricas. Sin embargo, EIGRP multiplica por 256 los componentes de sus métricas para de esta forma lograr una granularidad más fina que IGRP.

EIGRP cuenta con cuatro componentes (ver Figura 3.1, tomada de [16]):

- Módulos Dependientes del Protocolo (*Protocol Dependent Modules*).
- RTP (*Reliable Transport Protocol*).
- Descubrimiento/Recuperación de Vecinos (*Neighbor Discovery/Recovery*).
- DUAL (*Diffusing Update Algorithm*).

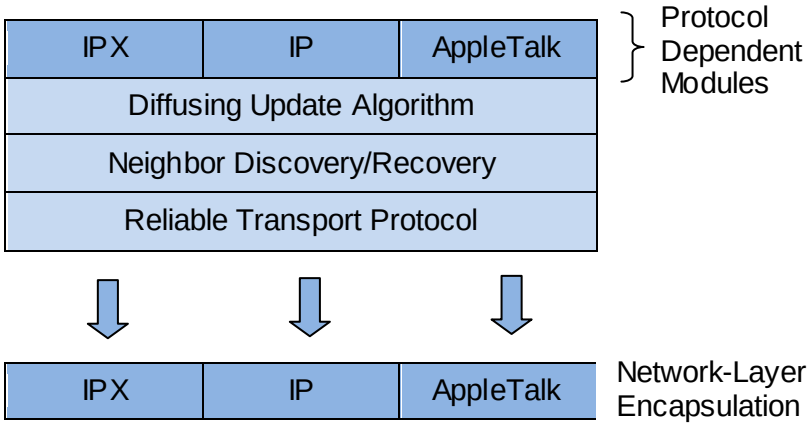

Figura 3.1: Componentes EIGRP

En la Figura 3.2 (tomada de [14]) se puede observar una visión general del funcionamiento de EIGRP y la interacción entre sus componentes principales.

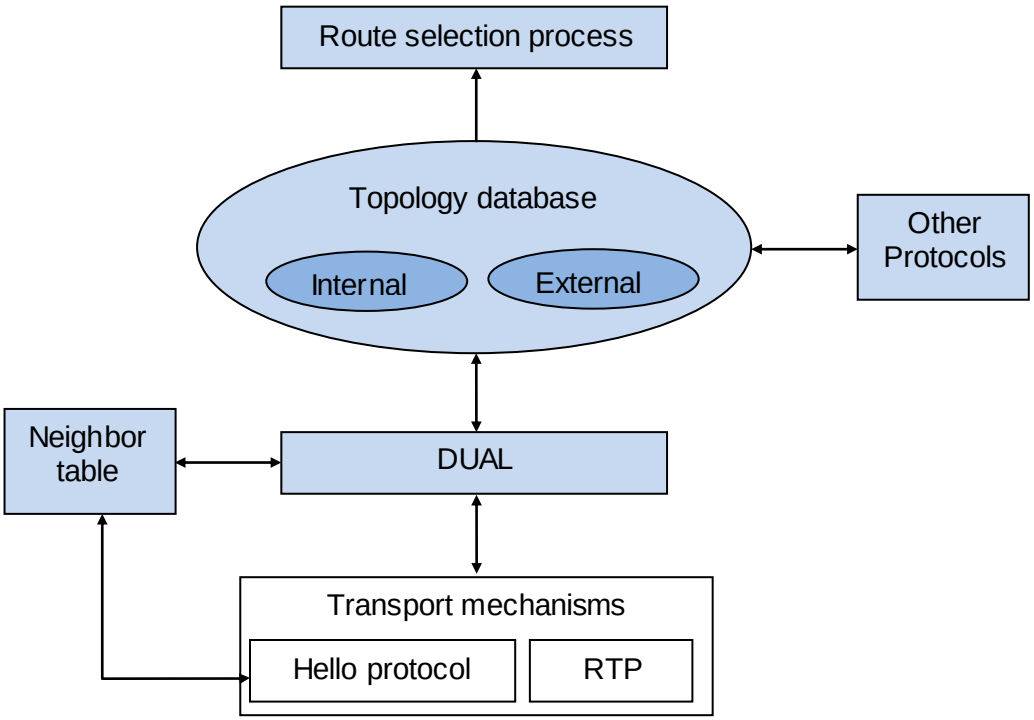

Figura 3.2: Funcionamiento de EIGRP

3.1.1 Módulos Dependientes del Protocolo

EIGRP implementa tres módulos, un módulo para IP, otro para IPX y otro para AppleTalk, los cuales son responsables de resolver las tareas de enrutamiento de algún protocolo en particular. Por ejemplo, el módulo de IPX es responsable de intercambiar información de enrutamiento entre redes IPX que utilicen EIGRP y debe también pasar información al DUAL. Adicionalmente, el módulo de IPX enviará y recibirá información SAP (*Service Advertising Protocol*)[7].

Como se puede observar en la Figura 3.1 el tráfico entre los módulos es encapsulado dentro de sus respectivos protocolos de capa de red. Por ejemplo, EIGRP para IPX es manejado por paquetes IPX.

EIGRP redistribuye información de manera automática cuando trabaja en conjunto con otros protocolos, por ejemplo:

- Cuando trabaja con IPX, redistribuye información de forma automática con IPX de RIP (*Routing Information Protocol*) [1] y NLSP (*NetWare Link Services Protocol*) [5].

[7] http://www.novell.com/documentation/nw4/docui/index.html#../cncptenu/data/h5g7f0on.html

- Cuando trabaja con AppleTalk, redistribuye de forma automática para AppleTalk de RTMP (*Routing Table Maintenance Protocol*) [24].
- Cuando trabaja con IP, redistribuye las rutas automáticamente para IGRP, siempre y cuando ambos sistemas compartan el mismo identificador de Sistema Autónomo (AS, *Autonomous System*).

EIGRP utiliza diferentes paquetes y mantiene separadas las tablas de vecinos, topología y las de enrutamiento, para cada protocolo de capa de red (ver Figura 3.3, tomada de [2]).

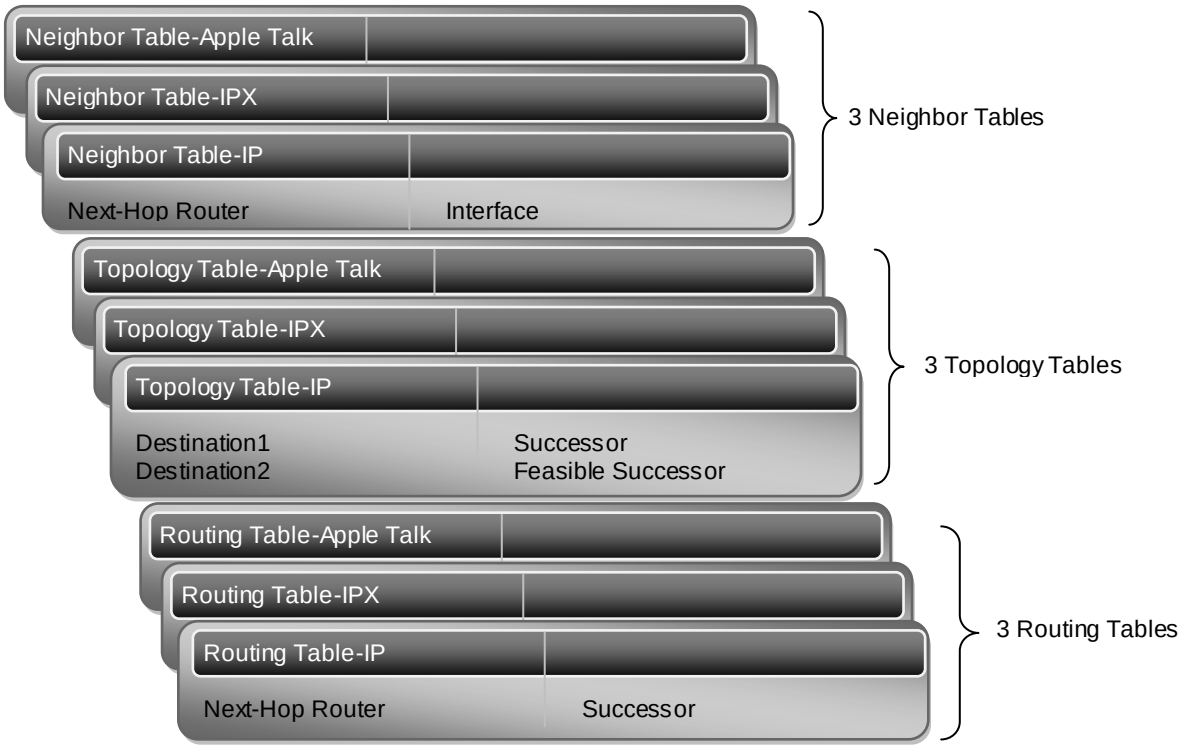

Figura 3.3: Tablas independientes para cada PDM

3.1.2 RTP (Reliable Transfer Protocol)

RTP administra el envío y recepción de los paquetes EIGRP de forma confiable dado que este protocolo garantiza que los paquetes serán entregados en orden. La entrega confiable se lleva a cabo utilizando el algoritmo propiedad de Cisco Systems llamado *Reliable Multicast*; dicho algoritmo utiliza la dirección reservada de clase D 224.0.0.10. Cada vecino que recibe un paquete multicast confiable está en la obligación de responderlo con un paquete de reconocimiento (ACK, *Acknowledgment*) unicast.

La entrega ordenada es asegurada incluyendo dos números de secuencia por paquete. Cada paquete incluye un número de secuencia asignado por el router origen, el cual es incrementado cada vez que el router envía un nuevo paquete. Además, el router origen coloca el número de secuencia del último paquete recibido del router destino en el campo ACK.

En algunos casos, RTP puede utilizar transmisiones no confiables, donde los reconocimientos no son requeridos y por ende los números de secuencia antes mencionados no deben ser incluidos en los paquetes.

EIGRP utiliza múltiples tipos de paquetes, todos ellos identificados con el número de protocolo 58 en la cabecera IP. Dichos paquetes se listan a continuación:

- *Hello*: Son utilizados por los vecinos para descubrir y recuperar adyacencias. Los paquetes *hello* son multicast y utilizan transmisiones no confiables (ver Figura 3.4, tomada de [2]).

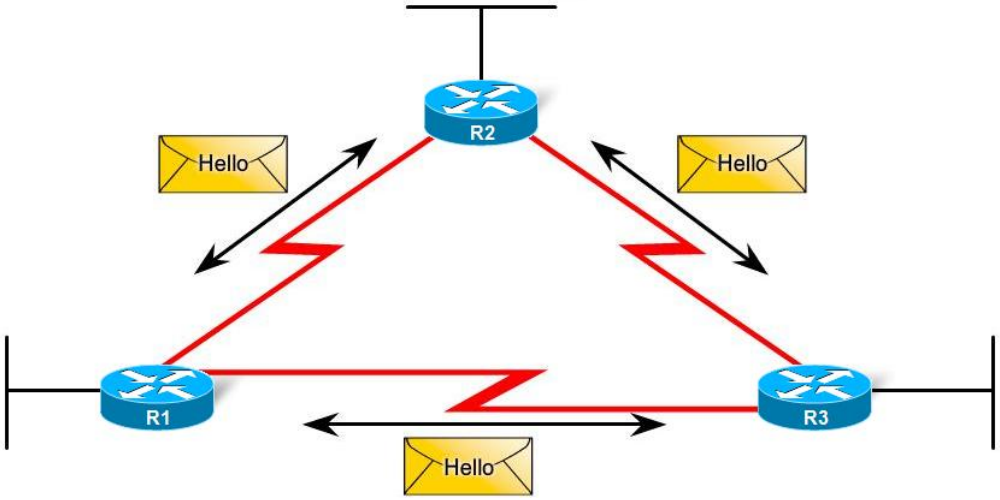

Figura 3.4: Paquetes *Hello*

- ACK (*Acknowledgment*): Son paquetes *hello* sin carga útil o sin datos. Los reconocimientos son siempre unicast y utilizan transmisiones no confiables.
- *Update*: Transmiten información de enrutamiento. A diferencia de las actualizaciones de RIP e IGRP, estos paquetes son transmitidos únicamente cuando son necesarios, conteniendo sólo la información pertinente y son enviados exclusivamente a los routers que requieran dicha información. Cuando las actualizaciones son requeridas por múltiples routers (por ejemplo, en caso de encontrarse un cambio de métrica o de topología), son enviados en multicast. Las actualizaciones utilizan siempre entregas confiables (ver Figura 3.5, tomada de [2]).

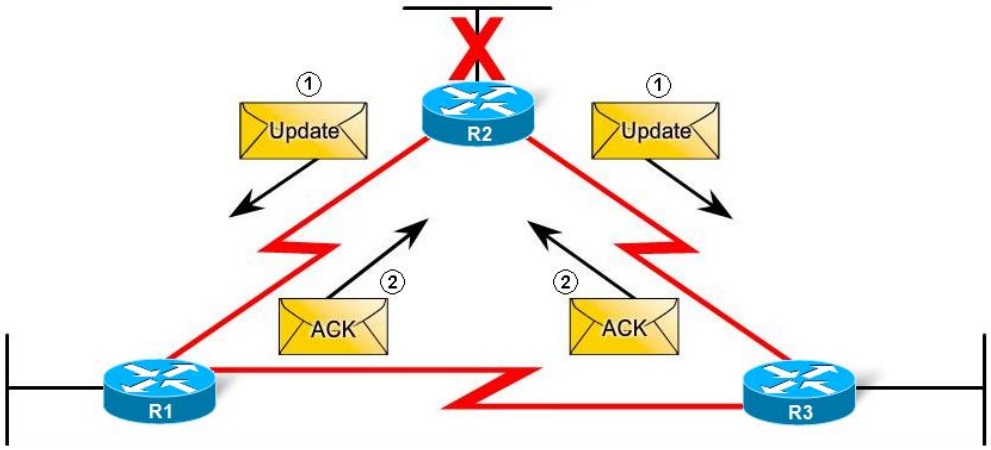

Figura 3.5: Paquetes *Update*

- *Queries* y *Replies*: Son utilizados por la máquina de estados finitos DUAL (ver sección 3.1.4) para administrar los *diffusing computations*. Los *queries* pueden ser, tanto multicast como unicast y, los *replies* son siempre unicast. Ambos trabajan con entregas confiables (ver Figura 3.6, tomada de [2]).

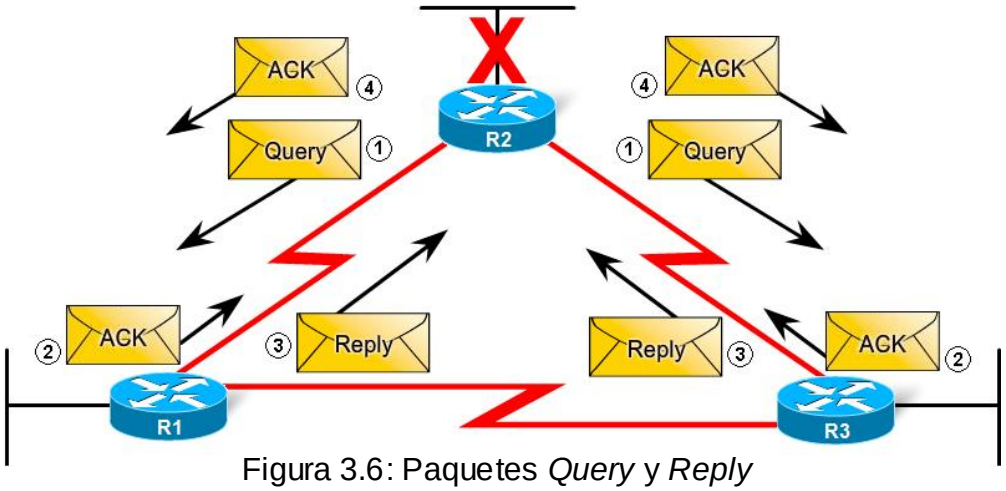

Figura 3.6: Paquetes *Query* y *Reply*

Cuando un router procesa un *query* de un vecino, las reglas que se muestran en la Tabla 3.1 aplican [8].

***Query* enviado por**	**Estado de la Ruta**	**Acción**
Vecino (no el *successor* actual)	Pasivo	Responder con información del *successor* actual
Successor	Pasivo	Iniciar una búsqueda para un nuevo *successor*, si es exitosa, responder con la nueva información, sino, marcar el destino como inalcanzable y enviar *queries* a todos los vecinos excepto al *successor* anterior
Cualquier vecino	No existe una ruta a través de este vecino antes de recibir el *query*	Responder con la mejor ruta conocida
Cualquier vecino	No conocido antes de recibir el *query*	Responder que el destino es inalcanzable
Vecino (no el *successor* actual)	Activo	Si no existe un *successor* actual para dichos destinos (lo que sucede normalmente), responder con destino inalcanzable
		Si existe algún *successor*, responder con la información de la ruta actual
Successor	Activo	Iniciar una búsqueda para un nuevo *successor*, si es exitosa, responder con la nueva información, sino, marcar el destino como inalcanzable y enviar *queries* a todos los vecinos excepto al *successor* anterior

Tabla 3.1: Reglas de procesamiento de queries

- *Requests*: Es un tipo de paquete que originalmente estaba destinado para ser utilizado en servidores de rutas, sin embargo, estos paquetes nunca fueron utilizados.

Si algún paquete trabaja con multicast confiable y el ACK respectivo no es recibido, el paquete será retransmitido utilizando unicast al vecino correspondiente. Si un ACK no es recibido luego de 16 retransmisiones, el vecino será declarado muerto o inactivo.

El tiempo de espera por un ACK antes de cambiar el envío multicast a unicast es especificado por el temporizador denominado *multicast flow timer*. El tiempo entre los unicast subsecuentes es especificado por el RTO (*Retransmission Time Out*). Tanto el *multicast flow timer* como el RTO son calculados en cada vecino por el SRTT (*Smooth Round-Trip Time*). SRTT es el tiempo promedio entre la transmisión de un paquete a un vecino y la recepción del ACK correspondiente.

Números de Secuencia y *Acknowledgments*

Cada protocolo de transporte confiable debe emplear algún tipo de variante de secuenciación para poder detectar pérdida de paquetes, retransmisiones y poder manejar la reordenación de paquetes. Las técnicas de secuenciación más conocidas son:

- Secuenciar toda sesión de aplicación, por ejemplo, TCP (*Transmission Control Protocol*) [16].
- Secuenciar sólo el tráfico nodo a nodo, por ejemplo, LLC2 (*Logical Link Control type 2*) [15] o X.25 [17].
- Utilizar conteo de bytes como número de secuencias o cuenta de paquetes.

Ninguno de los métodos antes mencionados puede ser aplicado al protocolo EIGRP debido a que no trabajan en operaciones de tipo unicast y multicast simultáneas. Para implementar un protocolo de transporte de tipo unicast/multicast, se pueden utilizar uno de los siguientes enfoques:

- Utilizar distintos números de secuencia para flujos unicast punto a punto y para flujos multicast punto a interfaz.
- Utilizar los mismos números de secuencia para todos los paquetes, pero aceptar que los números de secuencia recibidos son no secuenciales.

Los diseñadores de EIGRP decidieron utilizar el segundo enfoque. Los números de secuencia EIGRP pudieron haber sido generados en base a interfaces o en base a routers pero, la segunda opción fue la elegida. Cada vez que un proceso EIGRP genera un nuevo paquete, el paquete lleva el próximo número de secuencia mayor. Estos paquetes de datos son, en condiciones genéricas, destinados a diferentes vecinos. Cada vecino debería entonces ver una cadena de paquetes no secuenciales llegando de cualquier router, haciendo inútiles los algoritmos tradicionales de ventanas y de retransmisión. La única posible solución es utilizar ventanas de tamaño uno; cada paquete recibido por un router debe ser reconocido individualmente.

RTP utiliza el conocido modo de operación *stop-and-wait* (o *ping-pong*). Cada paquete de datos es normalmente reconocido por un paquete ACK que lleva el mismo número de secuencia, pero también puede ser reconocido por otro paquete de datos unicast que viaja en dirección contraria (ACK *piggybacked*), el cual es generalmente un *reply* a un *query*, sin embargo, el ACK *piggybacked* puede ocurrir siempre que haya un paquete de datos unicast encolado para viajar en dirección contraria. Para permitir ambos tipos de reconocimientos, cada paquete EIGRP contiene dos campos (*sequence number* y *acknowledgment number*) que llevan los valores especificados en la Tabla 3.2.

Condición	***Sequence Number***	**ACK *Number***
Paquetes de datos enviados antes de recibir el primer paquete de un vecino	*Sequence Number* correspondiente	0
Paquetes de datos enviados después de recibir el primer paquete de un vecino	*Sequence Number* correspondiente	Último *sequence number* recibido del vecino
Paquete ACK	0	Último *sequence number* recibido del vecino
Paquete *Hello*	0	0

Tabla 3.2: Sequence Numbers y ACK Numbers

Retransmisiones y temporizadores de retransmisiones

Los números de secuencia y los números de reconocimiento o ACK son llevados en los paquetes EIGRP, permitiendo así que RTP pueda recuperarse de diversos tipos de pérdida de paquetes. El caso más sencillo es cuando se vence el temporizador de retransmisión antes de recibirse un ACK, donde se procede a retransmitir el paquete como se muestra en la Figura 3.7.

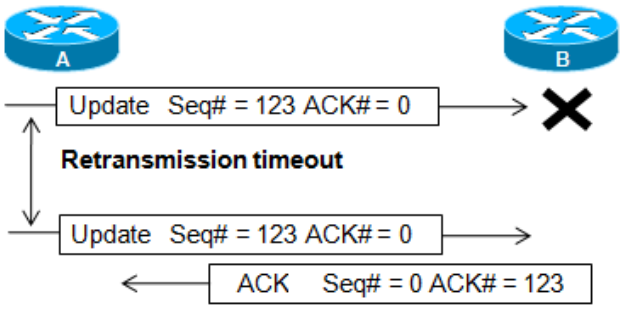

Figura 3.7: Recuperación de RTP luego de una pérdida de paquete

Si el ACK se pierde, el router origen no puede detectar dicha condición y maneja la excepción como si se hubiera perdido el paquete original, es decir, reenviando el paquete. El router destino puede detectar paquetes duplicados dado que siempre guarda el último *sequence number* recibido de cada vecino. El mismo descarta el duplicado y reenvía un ACK para el paquete de datos (ver

Figura 3.8).

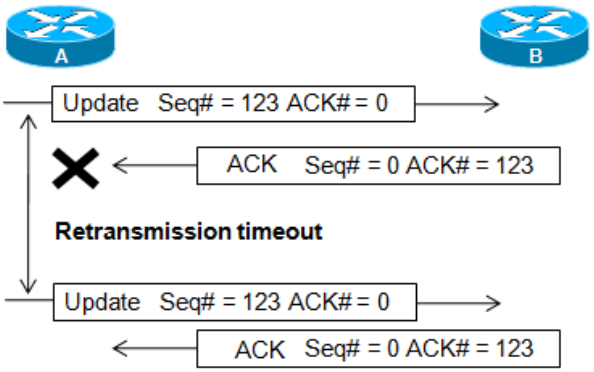

Figura 3.8: Recuperación de RTP luego de una pérdida de un ACK

Es crucial para todo protocolo de transporte mantener sus operaciones óptimas y eficientes, para ello es importante el valor que se escoge para los temporizadores de retransmisiones (*retransmission time out*). Si el temporizador tiene asociado un valor muy pequeño, el protocolo no utilizará de forma efectiva el ancho de banda debido a que generará demasiadas retransmisiones innecesarias. Si por el contrario, el temporizador tiene un valor asociado muy grande, la convergencia del protocolo de enrutamiento decaerá muy rápido. Para asegurar que las operaciones sean continuas y eficientes, todo protocolo de transporte debe ajustar constantemente sus

temporizadores de retransmisión para así ajustarse a cambios en las condiciones de la red y algunas otras variantes.
RTP calcula el RTT (*Round Trip Time*) en cada intercambio de paquetes. El RTT está definido como el intervalo entre el envío de un mensaje y la recepción del ACK del mismo. Luego de cada cálculo del RTT, RTP computa el SRTT (*Smoothed Round Trip Time*) para cada vecino utilizando la siguiente fórmula (tomada de [14]):

$$[4]\ SRTT_{actual} = SRTT_{anterior} * 0.8 + RTT * 0.2$$

A su vez, también es calculado el RTT para cada interfaz.

El RTO (*Retransmission Time Out*) se calcula de la siguiente manera (tomada de [14]):

$$[5]\ RTO_{inicial} = 6 * \max(SRTT, PacingInterval)$$

El *PacingInterval* es utilizado para prevenir que el RTO expire mientras que el paquete aun se encuentra en una cola de salida. El RTO es incrementado en un 50% después de cada retransmisión, como es mostrado en la siguiente fórmula (tomada de [14]):

$$[6]\ RTO_{actual} = RTO_{anterior} * (3/2)$$

El RTO no puede ser menor de 200 microsegundos o mayor a 5 segundos, por ello, debe darse que (tomada de [14]):

$$[7]\ RTO_{final} = \min(5000, \max(200, RTO_{calculado}))$$

3.1.3 Descubrimiento y Recuperación de Vecinos

Dado que los paquetes de *update* de EIGRP no son periódicos, es de especial importancia contar con un proceso que se encargue de descubrir y hacer seguimiento de los routers vecinos. En la mayoría de las redes, los paquetes *hello* son de tipo multicast y son enviados cada 5 segundos (propiedad configurable) menos un tiempo aleatorio que se utiliza para prevenir la sincronización. Estos paquetes no requieren de reconocimientos o ACKs.

Cuando un router recibe un paquete *hello* de un vecino, éste incluirá un tiempo de espera llamado *hold time* (configurable manualmente y cuyo valor por defecto es de 15 segundos) el cual indica el tiempo máximo que el router debe considerar activo a un vecino específico. En caso de que dicho temporizador se agote, el vecino es declarado como inalcanzable y DUAL es informado sobre la pérdida detectada.
La capacidad de EIGRP de poder detectar un vecino perdido en 15 segundos (a diferencia de en 180 segundos como sucedería en RIP o 256 segundos en IGRP) es un factor que contribuye a la rápida convergencia del protocolo.

La información de cada vecino es almacenada en una tabla de vecinos (ver Figura 3.9). Dicha tabla de vecinos se encuentra compuesta por:

- La dirección IP del nodo adyacente (*Address*) y la interfaz (*Interface*) por la cual los paquetes *hello* son recibidos.
- *Hold Time*, es decir, el tiempo en el que se considera un vecino vivo sin recibir paquetes *hello* de su parte.
- SRTT (*Smoothed Round-Trip Time*).
- *Uptime* (tiempo de actividad) el cual contabiliza el tiempo transcurrido desde que un vecino es almacenado en la tabla.
- RTO, es el tiempo, medido en milisegundos, que el router esperará por un reconocimiento o ACK de un paquete unicast enviado posterior a un paquete multicast. Si un *update* EIGRP, *query* o *reply* es enviado, una copia del mismo será encolada. En caso de que el RTO expire antes de que un ACK sea recibido, la copia previamente encolada es enviada.
- *Q Count* indica el número de paquetes encolados.
- *Seq Num* (número de secuencia) se refiere al último *update*, *query* o *reply* recibido. RTP hace seguimiento de estos números de secuencia para asegurar que los paquetes sean recibidos en el orden correcto.
- Por último, se almacena un índice según el orden en que los vecinos fueron aprendidos (*H*) [16].

```
IP-EIGRP neighbors for process 90
H   Address      Interface   Hold   Uptime    SRTT  RTO  Q   Seq
                             (sec)             (ms)      Cnt Num
3   10.1.2.2     Fa0/0       11     01:19:38   43   258  0   36
2   10.1.3.2     Se0/0/0     12     01:25:26   20   200  0   43
1   10.1.1.1     Se0/0/1     14     01:26:28    8   200  0   40
0   10.1.4.2     Fa0/1       11     01:26:39   34   204  0   46
```

Figura 3.9: Tabla de vecinos (*Neighbor Table*)

En líneas generales, el proceso de formación de vecinos EIGRP se basa en un mecanismo de *three way handshake* sobre un enlace unidireccional.

En la Figura 3.10 se puede observar dicho mecanismo. Cuando el router A recibe el primer paquete multicast *hello* generado por el router B, este es colocado en *pending state* o estado pendiente, acto seguido, el router A transmite un *update* unicast con el bit de inicialización (*init*) del campo *flags* (ver sección 3.1.5) encendido. Mientras que B se encuentre en estado *pending*, A no le enviará ningún *query* o información de enrutamiento.

Cuando B recibe el *update* con el bit *init* encendido, este envía un *update* con el bit *init* encendido de igual forma. El número de secuencia del *update* inicial es *piggybacked* dentro de este paquete (nunca es transmitido de forma individual), no hay forma de que A reciba un ACK para su *update* inicial sin recibir al mismo tiempo el *update* inicial de B.

Una vez que el *update* es recibido por A, este envía el paquete ACK correspondiente modifica el estado pendiente de B y se inicia el intercambio de información de topologías.
Si el ACK para A nunca es recibido, los paquetes *hellos* de B son ignorados mientras que A intenta retransmitir el *update* inicial. Eventualmente, el tiempo de espera de A se agotará y el proceso se iniciará nuevamente [4].

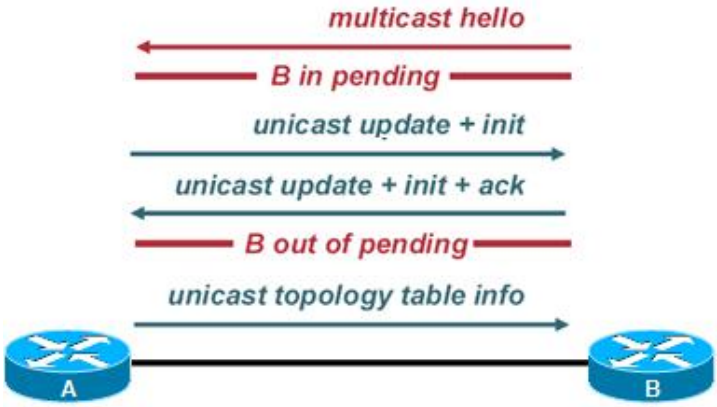

Figura 3.10: Proceso de formación de vecinos

3.1.4 DUAL (Difussing Update Algorithm)

DUAL es el mecanismo de recálculo de rutas para el protocolo EIGRP. DUAL dicta que, los ciclos (incluyendo los temporales) en las rutas son perjudiciales para el desempeño de una intranet. DUAL utiliza *diffusing computations* (propuesto por E.W. Dijkstra y C.S Scholten) con el fin de poder llevar a cabo un enrutamiento distribuido de distancias más cortas evitando los ciclos a toda costa.

Para que DUAL funcione de forma correcta, un protocolo de menor nivel debe asegurarse de que las siguientes condiciones sean cumplidas:

- Un nodo debe detectar en un tiempo finito la existencia de un nuevo vecino o la pérdida de conectividad con un vecino.
- Todos los mensajes transmitidos sobre un enlace son recibidos de forma correcta y en la secuencia apropiada en un tiempo finito.
- Todos los mensajes, cambios en el costo de un enlace, fallas de enlaces o notificaciones de nuevos vecinos deben ser procesados uno a la vez en un tiempo finito y en el orden que hayan sido detectados.

EIGRP utiliza el mecanismo de descubrimiento y recuperación de vecinos además del protocolo RTP para asegurar dichas precondiciones.

Antes de describir el funcionamiento de DUAL algunos conceptos deben ser aclarados.

Adyacencia:

En un principio, los routers utilizan paquetes *hello* para descubrir vecinos y para presentarse a ellos. Cuando un vecino es descubierto, EIGRP tratará de formar una

adyacencia con el mismo. Una adyacencia es un enlace virtual entre dos nodos por el cual se transmite información. Cuando las adyacencias han sido establecidas, el router recibirá actualizaciones de sus vecinos. Las actualizaciones contendrán todas las rutas conocidas por el router emisor además de las métricas respectivas. Para cada ruta, el router calculará una distancia basada en la distancia anunciada por el vecino y el costo del enlace hasta el vecino.

FD (*Feasible Distance*):

La métrica de menor peso calculada hacia cada destino será la FD para el mismo. Por ejemplo, un router puede ser informado de 3 rutas distintas para alcanzar la red 172.16.5.0 y, podrían calcularse métricas de 380672, 12381440 y 660868 para las rutas. En este caso, 380672 será la FD dado que es la de menor peso [16].

RD (*Reported Distance*):

Las RD son aquellas distancias a un destino que son reportadas a un router por sus vecinos. Cualquier ruta encontrada menor que la FD es considerada por DUAL como una ruta libre de ciclos, sin embargo, existen posibilidades de que el algoritmo etiquete una ruta como cíclica sin serlo pero jamás sucederá lo contrario [4].

FC (*Factible Condition*):

Es una condición que se cumple cuando la distancia anunciada (RD) de un vecino a un destino específico es menor que la FD del router.

***Feasible Successor*:**

Si un vecino anuncia una distancia a un destino que coincide con la FC, el vecino se convierte en un *feasible successor*. Por ejemplo, si la FD para alcanzar la *subnet* 172.16.5.0 es 380672 y un vecino anuncia una ruta para dicha *subnet* con una distancia de 355072 (RD), el vecino se convertirá en un *feasible successor*; en cambio, si el vecino anuncia una distancia de 380928 (RD) entonces no será satisfecha la FC y no será tomado en cuenta como posible *feasible successor*.

El concepto de *feasible successor* y de FC son primordiales para evitar los ciclos debido a que, los *feasible successors* siempre disminuyen las distancias por lo que un router nunca optará por elegir una ruta que lo lleve a él mismo dado que, se supone que dicha ruta debe ser mayor que la FD.

Cuando existan uno o más *feasible successors* para un destino, éstos serán almacenados en una tabla topológica (ver Figura 3.11), incluyendo los siguientes campos**:**

- El FD destino.
- Todos los *feasible successors*.
- Cada anuncio de distancia (RD) recibido de cada *feasible successor* a un destino específico.

- La distancia local calculada a un destino a través de cada *feasible successor* basado en las distancias anunciadas (RD) por los *successors* y el costo de los enlaces a dicho *successor*.
- La interfaz conectada a la red en la que cada *feasible successor* es encontrado (en realidad la interfaz no es explícitamente almacenada en la tabla topológica sino es guardada como un atributo del vecino).

El intercambio de tablas de topología puede apreciarse en la Figura 3.12 (tomada de [2]) donde, para cada ruta que A envía a B, B envía la misma de regreso para así asegurar que las dos tablas de rutas son coherentes (ver Sección 3.1.6).

Cuando un router termina de enviar su tabla, este envía un indicador llamado *end of table indicator* [4].

```
Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - Reply status

P 172.16.252.0/24, 1 successors, FD is 2681856
         via 172.16.250.2 (2681856/2169856), Serial0
         via 172.16.251.2 (46738176/2169856), Serial1
P 172.16.250.0/24, 1 successors, FD is 2169856
         via Connected, Serial0
P 172.16.251.0/24, 1 successors, FD is 46226176
         via Connected, Serial1
P 172.16.50.0/24, 1 successors, FD is 2195456
         via 172.16.250.2 (2195456/281600), Serial0
P 172.16.1.0/24, 1 successors, FD is 128256
         via Connected, Ethernet0
P 172.16.100.0/24, 1 successors, FD is 2707456
         via 172.16.250.2 (2707456/2195456), Serial0
         via 172.16.251.2 (46251776/281600), Serial1
```

Figura 3.11: Tabla topológica (*Topological Table*)

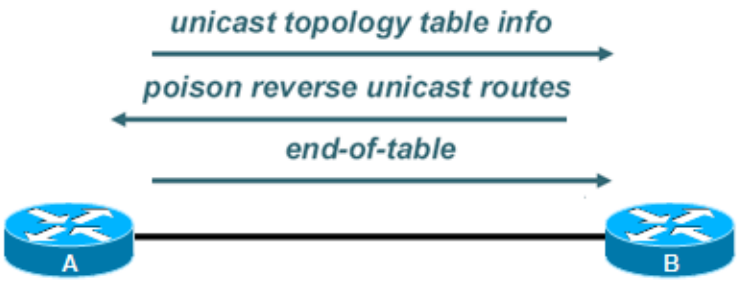

Figura 3.12: Intercambio de tablas de topología

La información de la tabla topológica puede ser insertada o actualizada cuando:

- Un paquete *update* sea recibido con un *delay* distinto de infinito.
- Un paquete *reply* sea recibido con un *delay* distinto de infinito.
- Una *subnet* directamente conectada a la red EIGRP se vuelve activa.

La información en la tabla topológica puede ser actualizada (pero no insertada) cuando un *query* con *delay* igual a infinito es recibido de un vecino. Finalmente, las entradas en la tabla pueden ser eliminadas cuando:

- Una *subnet* directamente conectada se vuelve inalcanzable (problemas de capa 1, capa 2 o la interfaz fue inhabilitada por un operador).
- Un *update*, *query* o *reply* es recibido con *delay* infinito.
- Un vecino es declarado muerto.

***Successor*:**

Para cada destino listado en la tabla topológica, la ruta con la menor métrica es elegida y colocada en la tabla de enrutamiento (ver Figura 3.13). El vecino que anunció dicha ruta se convierte en un *successor* o, el próximo salto por el cual los paquetes serán enviados para ese destino.

Las mejores rutas no son copiadas automáticamente en la tabla de enrutamiento, primero se debe llevar a cabo un proceso de selección tomando en cuenta diferentes fuentes de enrutamiento. La distancia administrativa es utilizada para dicho proceso de selección (ver Sección 3.4).

Generalmente, EIGRP sólo almacena rutas a los *successors* en la tabla de enrutamiento, siempre tomando en cuenta el proceso de balanceo de carga (ver Sección 3.1.7).

```
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate
default

Gateway of last resort is not set

     172.16.0.0/24 is subnetted, 6 subnets
D       172.16.252.0 [90/2681856] via 172.16.250.2, 00:18:54, Ethernet0/0
C       172.16.250.0 is directly connected, Ethernet0/0
C       172.16.251.0 is directly connected, Ethernet0/1
D       172.16.50.0 [90/2195456] via 172.16.250.2, 00:18:54, Ethernet0/0
C       172.16.1.0 is directly connected, Loopback0
D       172.16.100.0 [90/2707456] via 172.16.250.2, 00:18:54, Ethernet0/0
C    192.168.1.0/24 is directly connected, Loopback1
D       10.1.4.0 [90/33280] via 10.1.5.2, 00:19:08, FastEthernet0/0
```

Figura 3.13: Tabla de enrutamiento (*Routing Table*)

Por ejemplo, observando la tabla topológica de R4 (ver Figura 3.16) se puede observar el estado de la red en caso de presentarse el escenario de la Figura 3.14.

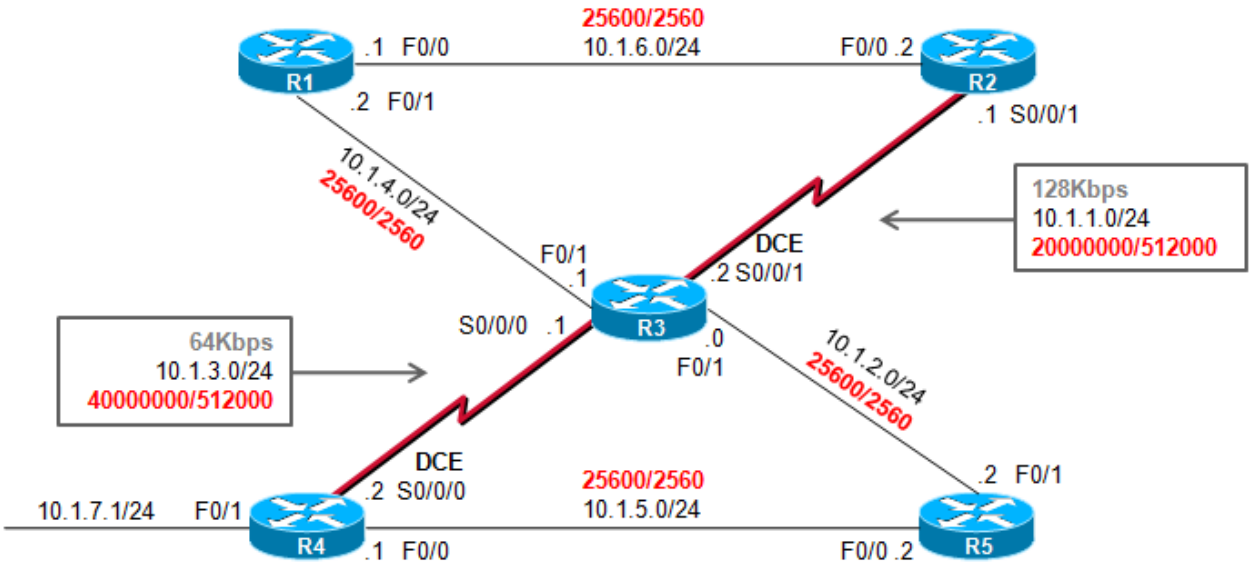

Figura 3.14: Escenario EIGRP

Adicionalmente, en la Figura 3.15 se muestran las interfaces de R4 las cuales se encuentran configuradas para manejar tráfico EIGRP. Utilizando dicha información, la cual puede verse de forma gráfica en la Figura 3.14, se puede observar que R4 cuenta con tres interfaces EIGRP, dos de tipo *Fast Ethernet* (F0/0 y F0/1) y una *Serial* (S0/0/0).

```
IP-EIGRP interfaces for process 90

                   Xmit Queue   Mean   Pacing Time   Multicast   Pending
Interface  Peers   Un/Reliable  SRTT   Un/Reliable   Flow Timer   Routes
Fa0/1        0        0/0         0       0/1            0           0
Se0/0/0      1        0/0        24      10/380        476           0
Fa0/0        1        0/0         1       0/1           50           0
```

Figura 3.15: Interfaces EIGRP de R4

Para comprender la aparición de los valores de las métricas en la tabla topológica (ver Figura 3.16) de R4, primero debe estudiarse la forma en la que los routers vecinos realizaron los cálculos de métricas hacia la red destino, en este caso la 10.1.6.0.

Debido a que R1 y R2 se encuentran directamente conectados a la *subnet* destino a través de interfaces *Fast Ethernet*, ambos anuncian una métrica de 28160, cantidad que se obtiene de aplicar la fórmula de cálculo de métricas compuestas: máximo(AnchosDeBanda)+∑delays, por ende, el valor obtenido es:

máximo(25600) + 2560 = 28160.

Luego de que R3 recibe dichas RD, el router debe calcular su FD. Para su interfaz F0/1, la FD obtenida es 30720, es decir:

máximo(25600, 25600) + 2560 + 2560 = 30720.

En cambio, para la interfaz S0/0/1 la FD es 20514560, es decir:

máximo(25600, 20000000) + 2560 + 512000 = 20514560.

A continuación, R3 anuncia la menor FD calculada (30720) hacia la *subnet* 10.1.6.0 a sus vecinos.

Seguidamente, R5 realiza los cálculos pertinentes para obtener su respectiva FD para la *subnet* destino y el valor que se obtiene es 33280, es decir:

máximo(25600, 25600) + 5120 + 2560 = 33280.

Como consecuencia de los cálculos descritos anteriormente, R4 recibe entonces dos RD para llegar a la *subnet* 10.1.6.0, la de R3 y la de R5. La FD que R4 obtiene utilizando la RD de R3 es 40517120 y a través de R5 la FD es 35840. Por ello, R4 toma a R5 como *successor* y coloca a R3 como *feasible successor* (dado que cumple con la *Feasible Condition*).

Entonces, puede verse que la ruta que R4 seguirá para alcanzar la *subnet* 10.1.6.0 es la siguiente: R4, R5, R3, R1.

Los *feasible successors* son importantes porque reducen el número de *diffusing computations* y por ello aumentan el desempeño. Además, los *successors* contribuyen a reducir los tiempos de convergencia. En caso de que un enlace a un *successor* falle o que el costo de dicho enlace supere el FD, el router primero tratará de utilizar otro *feasible successor* que se encuentre en la tabla correspondiente. El router iniciará *diffusing computations* de rutas sólo en caso de que no exista un *feasible successors* en dicha tabla.

Finalmente, puede observarse en la tabla de enrutamiento de R4 para el caso del escenario antes descrito. En ella puede verse que efectivamente el *successor* para la *subnet* 10.1.6.0 es 10.1.5.2 a través de la interfaz *Fast Ethernet 0/0*, es decir R5.

Operación de DUAL en caso de añadir una nueva ruta

Cuando una nueva *subnet* se vuelve alcanzable por un router, éste debe informar a sus vecinos (mediantes *updates*). El proceso para enviar *updates* es:

- Los *updates* individuales son encolados en la interfaz.
- Los *updates* son empaquetados y enviados cuando la interfaz se encuentra lista para enviar tráfico EIGRP.
- El paquete es enviado a cada vecino alcanzable por dicha interfaz.

```
IP-EIGRP Topology Table for AS(90)/ID(10.1.7.1)

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

P 10.1.3.0/24, 1 successors, FD is 40512000
        via Connected, Serial0/0/0
P 10.1.2.0/24, 1 successors, FD is 30720
        via 10.1.5.2 (30720/28160), FastEthernet0/0
        via 10.1.3.1 (40514560/28160), Serial0/0/0
P 10.1.1.0/24, 1 successors, FD is 20517120
        via 10.1.5.2 (20517120/20514560), FastEthernet0/0
        via 10.1.3.1 (41024000/20512000), Serial0/0/0
P 10.1.7.0/24, 1 successors, FD is 28160
        via Connected, FastEthernet0/1
P 10.1.6.0/24, 1 successors, FD is 35840
        via 10.1.5.2 (35840/33280), FastEthernet0/0
        via 10.1.3.1 (40517120/30720), Serial0/0/0
P 10.1.5.0/24, 1 successors, FD is 28160
        via Connected, FastEthernet0/0
P 10.1.4.0/24, 1 successors, FD is 33280
        via 10.1.5.2 (33280/30720), FastEthernet0/0
        via 10.1.3.1 (40514560/28160), Serial0/0/0
```

Figura 3.16: Tabla topológica de R4

```
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS
       level-2
       ia - IS-IS inter area, * - candidate default, U - per-user
       static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 7 subnets
C       10.1.3.0 is directly connected, Serial0/0/0
D       10.1.2.0 [90/30720] via 10.1.5.2, 00:19:14, FastEthernet0/0
D       10.1.1.0 [90/20517120] via 10.1.5.2, 00:19:08, FastEthernet0/0
C       10.1.7.0 is directly connected, FastEthernet0/1
D       10.1.6.0 [90/35840] via 10.1.5.2, 00:19:08, FastEthernet0/0
C       10.1.5.0 is directly connected, FastEthernet0/0
D       10.1.4.0 [90/33280] via 10.1.5.2, 00:19:08, FastEthernet0/0
```

Figura 3.17: Tabla de enrutamiento de R4

Existe una variante del común paquete *update* denominado: *poison update* o *poison reverse* (ver Sección 3.1.6). La finalidad de estos paquetes es evitar ciclos. En este caso, el campo *delay* del paquete *update* es colocado en infinito.

Operación de DUAL en caso de detectar una pérdida de una ruta

Cuando se detecta una pérdida de una ruta, EIGRP reporta la misma utilizando un *update* con el campo *delay* colocado a infinito.

En caso de que se pierda un enlace a una *subnet*, se deben eliminar todos los nodos que en algún momento fueron alcanzables por dicho enlace.

En resumen, se pueden enumerar tres reglas básicas que rigen el funcionamiento de DUAL, éstas son:

- Cuando un router elige a un nuevo *successor*, este envía su RD.
- Cuando un router elige a un *successor*, éste envía un *poison reverse* a su *successor*.
- Un *poison update* es enviado a todos los vecinos por la cual el *successor* es alcanzable, a menos que el *split horizon* (ver Sección 3.1.6) se encuentre deshabilitado, en cuyo caso sólo es enviado al *successor*.

Máquina de Estados Finitos DUAL

Un router reevaluará su lista de *feasible successors* para una ruta en cualquier momento que un evento de entrada (*input event*) ocurra. Un evento de entrada puede ser:

- Un cambio en el costo de un enlace directo.
- Un cambio en el estado (activo o inactivo) de un enlace directo.
- La recepción de un paquete *update*.
- La recepción de un paquete *query*.
- La recepción de un paquete *reply*.

El primer paso para la reevaluación es recalcular localmente la distancia hacia el destino para cada uno de los *feasible successors*. Los posibles resultados son:

- Si el *feasible successor* con la menor distancia es diferente del *successor* existente, el *feasible successor* será entonces el nuevo *successor*.
- Si la nueva distancia es menor que la FD, la misma será actualizada.
- Si la nueva distancia es diferente de la distancia existente se enviarán notificaciones a todos los vecinos.

Mientras que un router realiza cálculos locales, la ruta se mantiene en estado pasivo. Si un *feasible successor* es encontrado, una actualización es enviada a todos los vecinos y el estado no cambia.

Si un *feasible successor* no puede ser encontrado en la tabla topológica, el router iniciará *diffusing computations* y la ruta será cambiada a estado activo. Hasta que los

diffusing computations no sean completados y la ruta no sea llevada a estado pasivo nuevamente, el router no podrá:

- Cambiar la ruta del *successor*.
- Cambiar la distancia que hasta los momentos el mismo ha anunciado.
- Cambiar la FD de la ruta.
- Iniciar *diffusing computations* para la ruta.

En general, los *diffusing computations* se llevan a cabo siempre y cuando ocurra que:

- Se pierde la ruta actual y no se encuentra una ruta alterna.
- Se pierde la ruta actual y la nueva encontrada incluye al *successor* que causó un aumento de métrica.
- Se pierde la ruta actual y la nueva encontrada no incluye a un *feasible successor*.

El proceso de *diffusing computations* (ver Figura 3.18) es el siguiente:

- La ruta es colocada en estado activo, para de esta forma evitar ciclos.
- Se inicializa un temporizador (*active timer*) para asegurar que la red convergerá en un tiempo razonable.
- Se crea una tabla (*reply-status*) para hacer un seguimiento de las respuestas (*reply*) esperadas.
- Se envían *queries* multicast a todos los vecinos (éstos incluyen la mejor distancia temporal calculada hacia el destino o el valor infinito, según el caso).
- Se recolectan las respuestas de todos los vecinos y se almacenan en las tablas de topología correspondientes. Adicionalmente, se hacen las actualizaciones pertinentes en la tabla de *reply-status* creada anteriormente.
- La mejor respuesta es seleccionada y la nueva mejor ruta es almacenada en la tabla de enrutamiento.
- En caso de ser necesario, se envía un *update* a los vecinos [14].

Luego de recibir la consulta, los vecinos llevarán a cabo sus propios cálculos (ver Tabla 3.3).

Condición	Acción
Ruta no presente en la tabla topológica	Enviar *reply* con métrica igual a infinito
Ruta ya marcada como activa	Enviar *reply* con la mejor ruta (puede ser infinito) y se detiene el proceso
Query recibido de un no *successor*	Enviar *reply* con la mejor ruta actual
Query recibido solo del *successor*, no existen más vecinos	Enviar *reply* con métrica igual a infinito
Query recibido del *successor*	Seleccionar la mejor nueva ruta. Si ésta pasa por un *feasible successor*, responder con la nueva mejor ruta, sino, extender la *diffusing computation*

Tabla 3.3: Acciones tomadas por los routers al recibir un *query* EIGRP

Para cada vecino al cual le haya sido enviado un mensaje *query*, el router originador de estos mensajes encenderá una bandera de estatus (*status flag* r) para hacer seguimiento a todos los *queries* enviados. Los *diffusing computations* son completados cuando se hayan recibido los *replies* de cada *query* enviado.

En algunos casos un router no recibe un *reply* para cada *query* que envía. Por ejemplo, esto puede ocurrir en redes muy amplias con cualidades como bajo ancho de banda y pobre calidad de enlace. En caso de que el temporizador (*active timer*, inicializado en tres minutos por defecto) se agote antes de que todos los *replies* sean recibidos, la ruta será declarada como SIA (*Stuck-in-active*). El o los vecinos que no hayan respondido serán eliminados de la tabla de vecinos y, los cálculos dictarán que el vecino ha respondido con una métrica infinita.

Cuando los *diffusing computations* finalizan, el router origen colocará la FD en infinito para asegurar que cada router que responda los *queries* con distancias finitas al nodo destino, tenga oportunidad de convertirse en un *feasible successor*. Para cada uno de los *replies* recibidos, una métrica es calculada basada en la distancia anunciada en el *reply* más el costo del enlace hacia el vecino que envió el anuncio. Un *successor* es seleccionado basado en la menor métrica y, ese valor es asignado al FD. Cualquier *feasible successor* que no satisfaga la FC (*Feasible Condition*) para la nueva FD (*Feasible Distance*) será eliminado de la tabla topológica. Es importante destacar que un *successor* no es elegido hasta que todos los *replies* han sido recibidos.

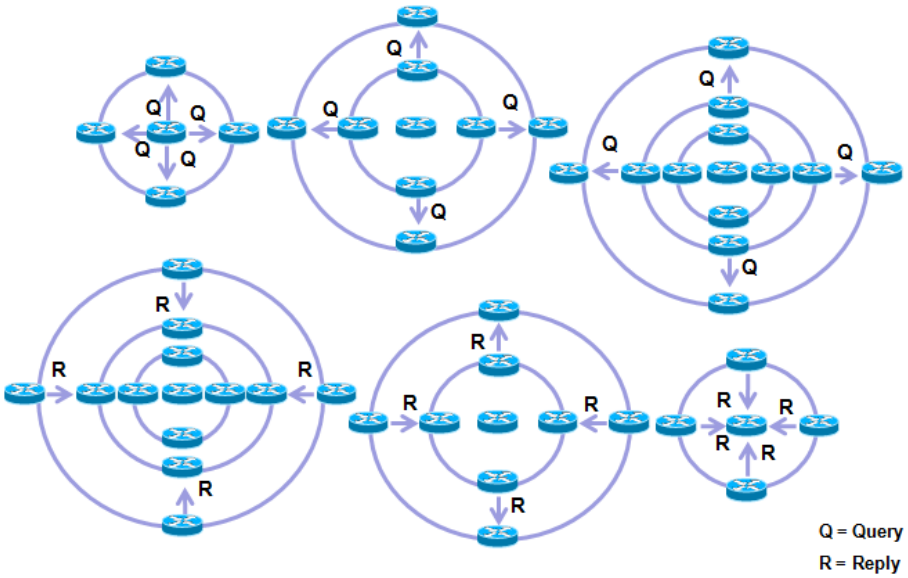

Figura 3.18: *Diffusing computations*

La Tabla 3.4 especifica los eventos de entrada que la máquina DUAL maneja, además, en la Figura 3.19 se puede observar por completo la configuración de estados de DUAL [16].

Evento de Entrada	Descripción
IE1	Cualquier evento por el cual el FC sea satisfecho o el destino sea inalcanzable
IE2	*Queries* recibidos por un *successor*; FC no satisfecho
IE3	Cualquier evento distinto de un *query* recibido de un *successor*; FC no satisfecho
IE4	Cualquier evento distinto a la recepción de un último *reply* o un *query* recibido de un *successor*
IE5	Cualquier evento distinto de un último *reply*, un *query* de un *successor* o un aumento en la distancia a un destino
IE6	Evento distinto de recepción de un último *reply*
IE7	Evento distinto a la recepción de un último *reply* o un aumento en la distancia a un destino
IE8	Aumento en la distancia a un destino
IE9	Último *reply* recibido; FC no coincide con el FD actual
IE10	Query o consulta recibida de un *successor*
IE11	Último *reply* recibido; FC coincide con el FD actual
IE12	Último *reply* recibido; se le asocia el valor de infinito al FD

Tabla 3.4: Eventos de entrada de DUAL

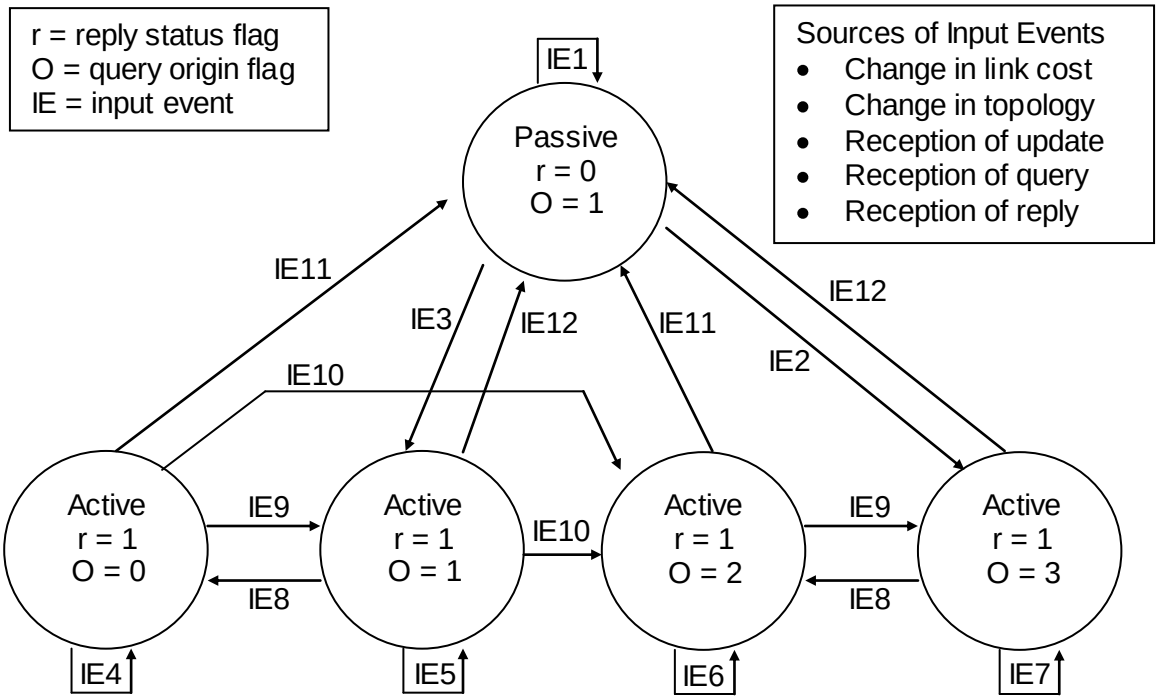

Figura 3.19: Máquina de estados finitos DUAL

3.1.5 Formato de los Paquetes EIGRP

La porción de datos en un mensaje EIGRP (ver Figura 3.20) es encapsulada en un paquete. Este campo de datos es llamado TLV (*Type/Length/Value*).

La cabecera EIGRP es incluida en todos los paquetes del protocolo, dicha cabecera en con su respectivo TLV son encapsulados en un paquete IP. En la cabecera del paquete IP se especifica el número de protocolo (EIGRP) y la dirección destino puede ser un dirección unicast o la dirección multicast (224.0.0.10).

En caso de que el paquete EIGRP sea encapsulado en una trama Ethernet, la dirección MAC multicast asociada a 224.0.0.10 es: 01-00-5E-00-00-0A [2].

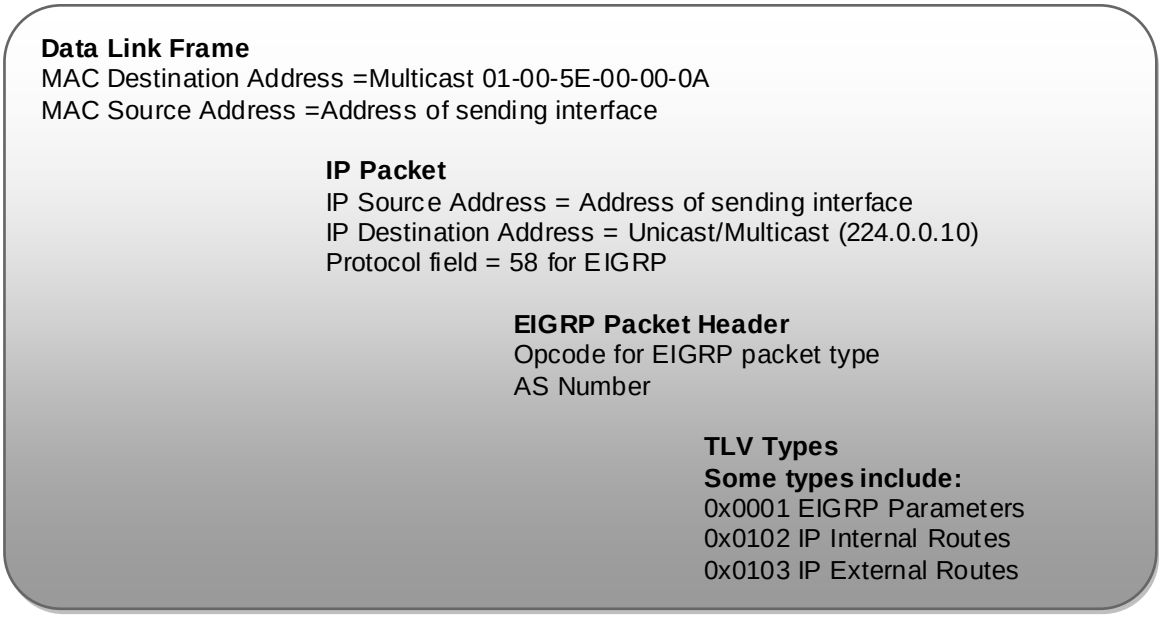

Figura 3.20: Paquete EIGRP

La cabecera EIGRP

La Figura 3.21 (tomada de [14]) muestra la cabecera EIGRP que se encuentra al inicio de cada paquete del protocolo.

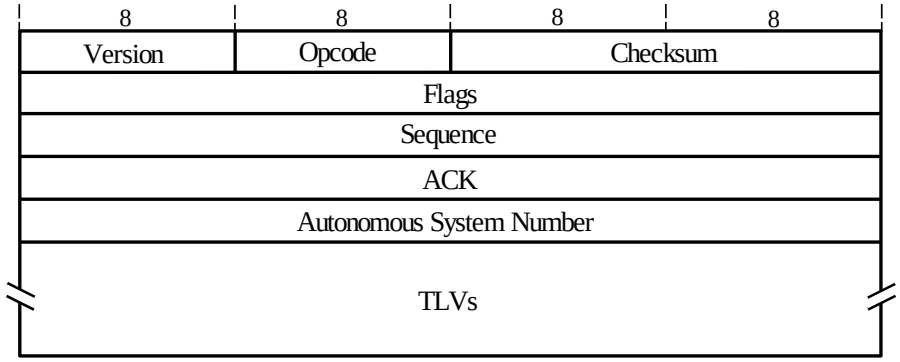

Figura 3.21: Cabecera EIGRP

- *Version:* especifica la versión particular de proceso EIGRP.
- *Opcode:* indica el tipo de paquete EIGRP, como se muestra en la Tabla 3.5.

Opcode	Tipo
1	*Update*
3	*Query*
4	*Reply*
5	*Hello*
6	IPX SAP

Tabla 3.5: Tipos de paquetes EIGRP

- *Checksum*: es el campo de verificación estándar de IP. Es calculado para todo el paquete EIGRP, excluyendo la cabecera IP.
- *Flags*: incluye sólo dos banderas. El bit más a la derecha (0x00000001) es *Init* que, cuando es encendido indica que las rutas que se transportan son las primeras en una nueva relación con un vecino. El segundo bit (0x00000002) es llamado *Conditional Receive*, y es utilizado en el algoritmo *Reliable Multicasting* propiedad de Cisco Systems.
- *Sequence*: es un número de secuencia de 32 bits utilizado por RTP.
- ACK: es un número de asentamiento de 32 bits que referencia al último paquete recibido de un vecino específico. Un paquete *hello* que lleve un número distinto de cero en el campo de ACK será tomado como un paquete ACK en vez de uno *hello*. Los ACKs nunca son paquetes multicast.
- *Autonomous System Number*: es el número de identificación del dominio EIGRP.

Luego de la cabecera, se encuentran los TLVs. Cada TLV incluye uno de dos tipos (interno o externo) de octetos mencionados en la Tabla 3.6, un campo de dos octetos especificando el tamaño del TLV y un campo variable cuyo formato es determinado por el tipo.

Número	Tipo de TLV
Tipos Generales de TLV	
0x001	Parámetros EIGRP
0x003	Secuencia
0x004	Versión del Software
0x005	Próxima Secuencia Multicast
Tipos de IP-Específicos TLV	
0x0102	Rutas Internas IP
0x0103	Rutas Externas IP
Tipos de AppleTalk TLV	
0x0202	Rutas Internas AppleTalk
0x0203	Rutas Externas AppleTalk
0x0204	Configuración del Cable AppleTalk
Tipos de IPX TLV	
0x0302	Rutas Internas IPX
0x0303	Rutas Externas IPX

Tabla 3.6: Tipos de TLV

Campos Generales TLV

Estos TLVs llevan consigo información de administración EIGRP y no información específica de algún protocolo. Los parámetros TLV, los cuales se usan para transmitir

pesos de métricas y el *hold time* (tiempo de espera), se muestran en la Figura 3.22 (tomada de [14]).

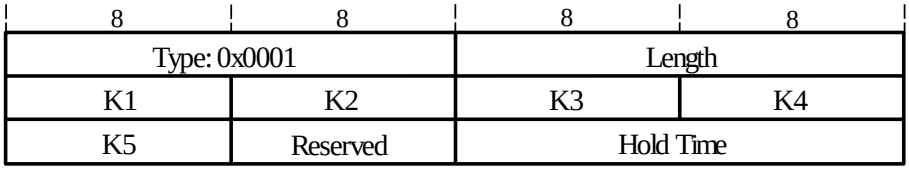

Figura 3.22: Parámetros TLV

Campos Específicos para IP de TLV

Cada ruta interna y externa TLV contiene una entrada para una ruta. Cada paquete *update*, *query* y *reply* contiene al menos una ruta TLV.

Las rutas TLV internas y externas incluyen información de las métricas para la ruta. Cabe destacar que las métricas utilizadas para EIGRP son las mismas utilizadas para IGRP pero multiplicadas por 256.

Rutas IP Internas TLV

Una ruta interna es una ruta a un destino dentro de un sistema autónomo EIGRP. El formato de una ruta interna TLV es mostrada en la Figura 3.23 (tomada de [14]).

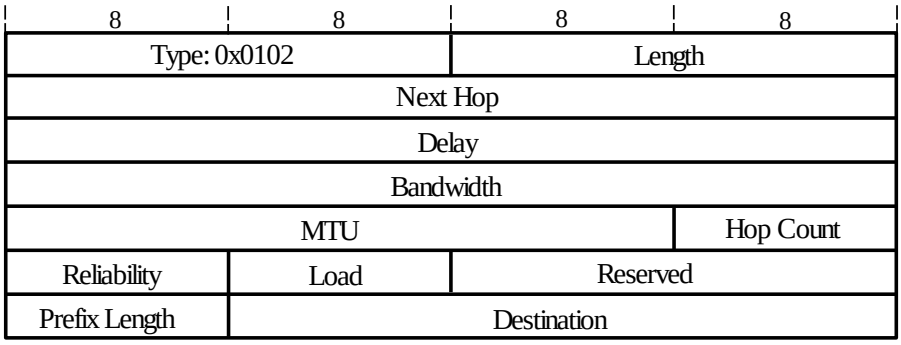

Figura 3.23: Rutas IP internas TLV

- *Bandwidth:* es calculado como: 256 * *bandwidth* IGRP, en otras palabras es 2560000000 dividido entre el menor *bandwidth* configurado a lo largo de la ruta.
- MTU: es el menor valor de la Unidad Máxima de Transmisión de todos los enlaces de la ruta hacia el destino.
- *Hop count*: es un número entre 0x01 y 0xFF indicando el número de saltos que se requieren para llegar al destino. Un router directamente conectado con una red particular anunciará un *hop count* de 0 (cero).
- *Realiability*: es un número entre 0x01 y 0xFF el cual refleja la cantidad total de errores salientes de las interfaces a lo largo de la ruta, calculado en un promedio exponencial de 5 minutos. El valor 0x01 indica que el enlace es poco fiable, mientras que 0xFF indica que el enlace es 100% fiable.
- *Load*: es también un número entre 0x01 y 0xFF y especifica la cantidad total de carga saliente de las interfaces a lo largo de la ruta, calculado en un promedio

exponencial de 5 minutos. El valor 0x01 indica que el enlace no recibe casi tráfico mientras que 0xFF indica un tráfico muy pesado.

- *Reserved*: no se utiliza y se encuentra siempre en 0x0000.
- *Prefix Length:* especifica el número de bits de red que se utilizan en la máscara.
- *Destination:* denota la dirección de la red destino. Este campo es de tamaño variable. Si la dirección a transmitir ocupa menos o más de tres octetos, el TLV será rellenado con ceros para completar los próximos cuatro octetos del paquete. Por ejemplo, si la dirección destino es 10.1.0.0/16, el campo *destination* será de dos octetos seguidos del valor 0x00. Si la dirección es 192.168.16.64/27, el campo *destination* será de cuatro octetos seguidos del valor 0x000000.

Rutas IP Externas TLV

Una ruta externa es una ruta que lleva a un destino fuera del demonio de enrutamiento EIGRP que ha sido redistribuido en EIGRP. La Figura 3.24 (tomada de [14]) muestra el formato de las rutas externas de este tipo.

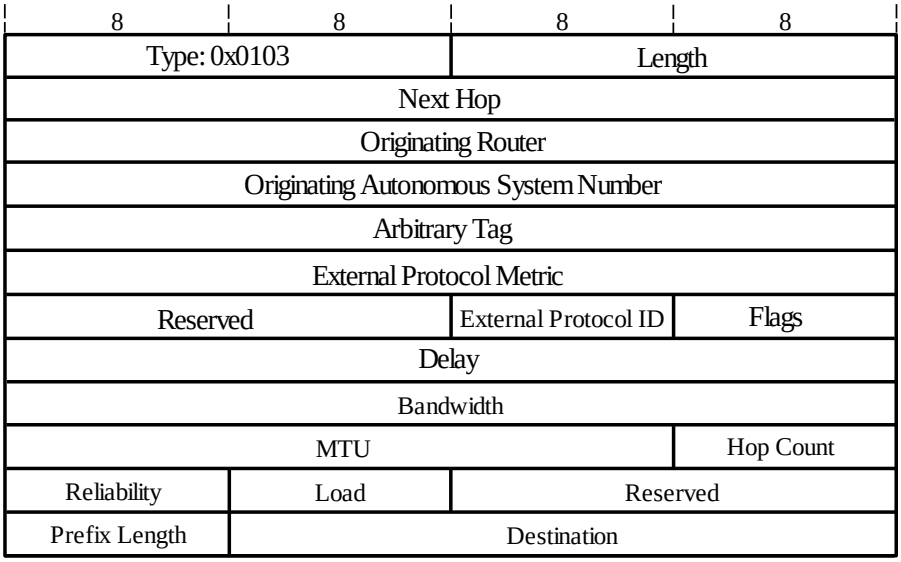

Figura 3.24: Rutas IP externas TLV

- *Originating Autonomous System Number:* se encarga de identificar el sistema autónomo del router que ha originado la ruta.
- *Arbitrary Tag:* puede ser utilizado para llevar consigo mapas de rutas.
- *External Protocol Metric*: es como su nombre lo indica, la métrica del protocolo externo.
- *External Protocol ID:* especifica el protocolo por el cual la ruta externa fue aprendida. La Tabla 3.7 presenta los posibles valores para este campo.

Código	Protocolo Externo
0x01	IGRP
0x02	EIGRP
0x03	Static Route
0x04	RIP
0x05	Hello
0x06	OSPF
0x07	IS-IS
0x08	EGP
0x09	BGP
0x0A	IDRP
0x0B	Connected Link

Tabla 3.7: Valores del campo de ID del protocolo externo

- *Flags:* está compuesto por dos banderas. El bit más a la derecha es encendido (0x01) cuando la ruta es externa, en cambio, si el segundo bit más a la derecha es encendido (0x02) entonces el paquete transporta una ruta por defecto. Una ruta por defecto (0.0.0.0/0 para IPv4) es la ruta de la red utilizada por el router cuando no existe ninguna otra ruta disponible para el destino especificado en el paquete IP.

El resto de los campos describen las métricas y la dirección destino.

3.1.6 Split Horizon y Poison Reverse

En algunas ocasiones, EIGRP utiliza *split horizon* para prevenir rutas cíclicas. La regla básica del s*plit horizon* es: nunca se debe anunciar una ruta por la interfaz por donde la misma fue aprendida.

Poison reverse es otra manera de evitar rutas cíclicas. En este caso la regla especifica: una vez que se ha aprendido una ruta a través de cierta interfaz, cuando deba de anunciarse como inalcanzable, debe hacerse por la misma interfaz.

EIGRP utiliza *split horizon* o anuncia rutas como inalcanzables cuando:

- Dos routers se encuentran en modo de inicio (*startup*), es decir, intercambiando tablas de topología por primera vez. Por cada ruta que un router reciba durante el período de *startup*, éste anunciará la misma entrada de vuelta a su vecino con la métrica infinita (ruta *poison*).
- Cuando se está anunciando un cambio en la tabla topológica. En caso de que se pierda un enlace, se supone que los routers deberían de informar de dicho cambio, a través de paquetes *update*, para de esa forma no invalidar las tablas de topología de sus vecinos; dado que en algunos escenarios el *split horizon* no lo permite, el mismo se apaga y se hace uso del *poison reverse*.
- Cuando se está enviando un *query*. Los *queries* sólo resultan con *split horizon* cuando un router recibe un *query* o un *update* desde el *successor* que esté utilizando como destino en el *query* [8].

3.1.7 Balanceo de Carga

EIGRP almacena las rutas con costos iguales en la tabla de enrutamiento la cual es utilizada por el router para distribuir la carga. El tipo de balanceo (por paquete o por destino) depende del tipo de *switching* realizado por el router. Sin embargo, EIGRP puede balancear carga sobre enlaces de distintos costos.

Supóngase que existen cuatro rutas para un destino específico, y las métricas de dichas rutas son:

- Ruta 1: 1100.
- Ruta 2: 1100.
- Ruta 3: 2000.
- Ruta 4: 4000.

Utilizando un protocolo de enrutamiento diferente de EIGRP, el router, coloca el tráfico en las rutas 1 y 2. Utilizando EIGRP, es posible configurar el router para tomar en cuenta las rutas 3 y 4, para ello debe especificarse un parámetro llamado varianza (*variance*), el cual es un multiplicando: el tráfico será colocado en cualquier enlace que tenga métrica menor que la mejor ruta multiplicado por dicha varianza. Por ejemplo, si se utiliza varianza 2, EIGRP sólo tomará en cuenta la ruta 3 (además de la ruta 1 y 2) dado que 1100 x 2 = 2200 > 2000.

Para balancear la carga, el router divide la métrica a través de cada ruta sobre la métrica más grande, redondea hacia abajo para obtener el entero más cercano y utiliza este número como contador de compartición de la carga. Por ejemplo, supóngase que los contadores de compartición son:

- Para las rutas 1 y 2: 4000/1100 = 3.
- Para la ruta 3: 4000/2000 = 2.
- Para la ruta 4: 4000/4000 = 1.

El router enviará los tres primeros paquetes por la ruta 1, los próximos tres paquetes por la ruta 2, los próximos dos paquetes por la ruta 3 y el último por la ruta 4. Finalmente, el router vuelve a enviar por la ruta 1, ruta 2, y así sucesivamente [8].

3.1.8 Summarization

Se define como el proceso de tomar dos o más redes IP y representarlas como una sola red IP.

Una dirección de *summarization* (ver Figura 3.25) representa un grupo numérico continuo de direcciones de red, conocido como súper-red. En la Figura 3.26 se puede observar el cálculo que se realiza para obtener la dirección de *summarization* del caso de la red de la Figura 3.25. El proceso es bastante simple, sólo deben buscarse los bits comunes para todas las direcciones de la red y enmascararlos.

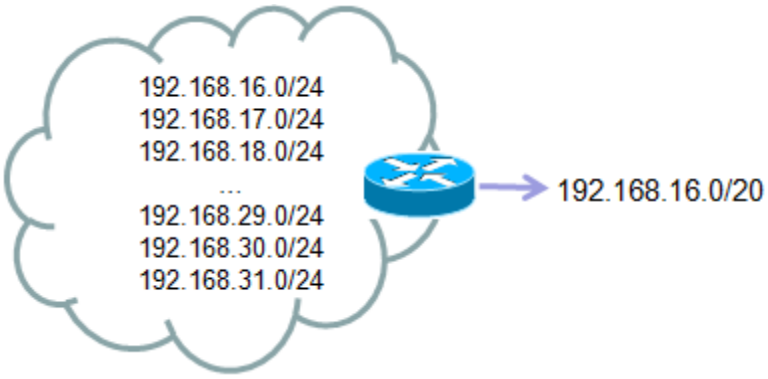

Figura 3.25: Ejemplo de *summarization*

```
11111111.11111111.11111111.00000000      =          24-bit mask
11111111.10101000.00010000.00000000      =          192.168.16.0/24
11000000.10101000.00010001.00000000      =          192.168.17.0/24
11000000.10101000.00010010.00000000      =          192.168.18.0/24
11000000.10101000.00010011.00000000      =          192.168.19.0/24
11000000.10101000.00010100.00000000      =          192.168.20.0/24
11000000.10101000.00010101.00000000      =          192.168.21.0/24
11000000.10101000.00010110.00000000      =          192.168.22.0/24
11000000.10101000.00010111.00000000      =          192.168.23.0/24
11000000.10101000.00011000.00000000      =          192.168.24.0/24
11000000.10101000.00011001.00000000      =          192.168.25.0/24
11000000.10101000.00011010.00000000      =          192.168.26.0/24
11000000.10101000.00011011.00000000      =          192.168.27.0/24
11000000.10101000.00011100.00000000      =          192.168.28.0/24
11000000.10101000.00011101.00000000      =          192.168.29.0/24
11000000.10101000.00011110.00000000      =          192.168.30.0/24
11000000.10101000.00011111.00000000      =          192.168.31.0/24
-------------------------------------------------------------------
11000000.10101000.00010000.00000000      =          192.168.16.0/24
```

Figura 3.26: Cálculo de una dirección de *summarization*

La ventaja de agrupar direcciones de red es el ahorro de los recursos de red, es decir, tanto el ancho de banda como los ciclos de CPU son conservados anunciando y procesando sólo algunas pocas rutas, pero, más importante aún, la memoria es conservada gracias a la reducción el tamaño de las tablas de enrutamiento.

El enrutamiento sin clase, VLSM (*Variable Length Subnet Mask*) y la *summarization* proveen el concepto de maximización de recursos construyendo jerarquías de direcciones. A diferencia de IGRP, EIGRP soporta todas esas estrategias de direccionamiento.

EIGRP utiliza dos formas de *summarization: summarization* automática y *summarization* manual.

Summarization Automática

Permite que EIGRP se comporte como un protocolo *classful* (como IGRP), siguiendo las siguientes reglas:

- Cuando se defina más de una red para un proceso EIGRP, el mismo debe crear una *subnet summarized* para cada una de las rutas definidas, tan rápido como al menos una de las *subnets* de una de las redes se encuentre en la tabla topológica.
- Una *subnet summarized* creada en el punto anterior, apunta a la interfaz Null 0 y debe tener la mínima métrica de todas las *subnets* de la red cubierta por la *subnet summarized*. La *subnet summarized* es almacenada también en la tabla de enrutamiento con una distancia administrativa de 5 (no configurable).
- Las *subnets summarized* en los puntos anteriores son ignoradas cuando el router envía paquetes *updates* en otras redes amplias IP.
- Las *subnets* que no pertenezcan a ninguna red listada en el proceso EIGRP, no serán *summarized* [14].

EIGRP realiza *summarization* automática cada vez que se cruza un borde entre dos redes amplias distintas. Por ejemplo, en la Figura 3.27, el router B anuncia sólo la red 10.0.0.0/8 al router A dado que la interfaz que el router B utiliza para alcanzar al router A se encuentra en una red amplia distinta.

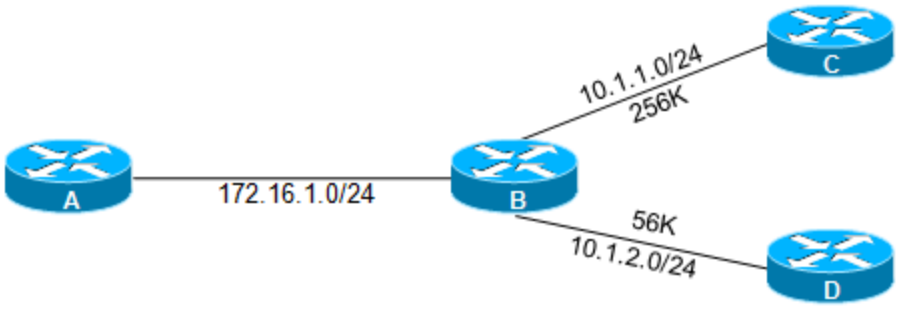

Figura 3.27: *Summarization* Automática

En conclusión, la *summarization* automática es definitivamente benéfica cuando las redes son migradas desde IGRP o RIP a EIGRP, esto se debe a que se retienen todas las propiedades de enrutamiento que la red tenía anteriormente. Esto garantiza que no se crearán ciclos de enrutamiento o cambios de flujos de tráfico después de la migración. La *summarization* automática introduce a su vez límites para los *queries* en redes donde existan redes amplias IP. Sin embargo, EIGRP perjudica a las redes donde existen redes amplias discontinuas.

Summarization Manual

Cuando la *summarization* automática no se adecua a las propiedades de la red, debido a discontinuidades en la red o debido a que el esquema de red no se encuentra compuesto por múltiples redes amplias, se utiliza la *summarization* manual, también conocido como *summarization* manual por interfaz.

Existen tres reglas que rigen este proceso, ellas son:

- Para cada rango de *summarization* configurado en una interfaz, EIGRP crea una *subnet summarized* tan rápido como una de las rutas del rango del *summarization* aparezca en la tabla topológica.
- La *subnet summarized* anteriormente debe apuntar a la interfaz Null 0 y debe tener la menor métrica de todas las rutas que comprende la *subnet summarized*.
- Las rutas cubiertas por la *subnet summarized* son ignoradas cuando se envían paquete *updates* por la interfaz por la que se configuró el rango de *summarization*. Los *updates* enviados sobre otras interfaces no son afectados [14].

EIGRP permite esta opción para rutas internas y externas utilizando cualquier bit de frontera de forma manual. Por ejemplo, en la Figura 3.28, el router B hace *summarization* de la red 192.1.1.0/24, 192.1.2.0/24 y la 192.1.3.0/24 en el bloque CIDR (*Classless Inter Domain Routing*) 192.1.0.0/22.

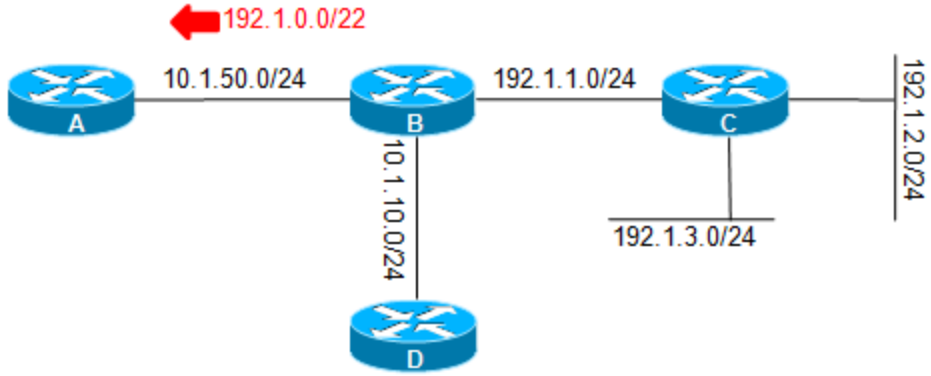

Figura 3.28: *Summarization* Manual

3.2 Autenticación MD5 con EIGRP

EIGRP admite la autenticación del router para proporcionar seguridad a las actualizaciones de la tabla de enrutamiento [22]. Cuando se utiliza la autenticación MD5, los routers se protegen de anunciar rutas a routers promiscuos no autorizados y de recibir rutas de ellos. La autenticación se define al nivel de interfaz y, por tanto, proporciona una selección discreta de las interfaces que deben utilizar la autenticación.

La autenticación queda determinada por la configuración de una clave específica para la interfaz. La técnica de autenticación MD5 es el único algoritmo válido de autenticación con EIGRP [13].

Luego de que una autenticación MD5 es configurada en una interfaz, cada paquete enviado por un router EIGRP por dicha interfaz es firmado con una huella digital MD5. Para cada paquete EIGRP recibido por una interfaz configurada con autenticación MD5, se debe verificar que la huella MD5 contenga el valor esperado.

MD5 es un algoritmo que toma un mensaje (un paquete EIGRP en este caso) y genera un valor *hash* de 128 bits con algunas propiedades que hacen que MD5 sea utilizado en implementaciones que requieren altos niveles de seguridad, algunas de ellas son:

- Cambiar un único bit en el mensaje original cambiará aproximadamente la mitad de los bits en la huella MD5 del mensaje.
- Es casi imposible generar otro mensaje que genere la misma huella MD5, por lo que tratar de crearlo es muy difícil.

El valor MD5 generado del paquete EIGRP es adjuntado a este mismo paquete y luego es enviado al nodo correspondiente. El router receptor puede verificar la integridad del paquete recalculando el valor MD5 y comparando con el valor del paquete recibido.

El proceso antes descrito no conlleva a la seguridad deseada porque cualquier intruso puede repetir los pasos llevados a cabo por el router emisor y generar paquetes forjados con las firmas correctas. Una clave secreta sólo conocida por el router emisor y el receptor debe ser utilizada para detener al intruso en sus intentos de quebrantar la comunicación. El proceso de intercambio seguro de información entre vecinos EIGRP puede ser resumido en los siguientes pasos (presentados gráficamente en la Figura 3.29, tomada de [14]):

- El router emisor genera la información EIGRP que va a enviarse.
- El *hash* MD5 es calculado utilizando la información EIGRP y la clave secreta compartida.
- El *hash* MD5 resultante es adjuntado al paquete y es enviado a su destino correspondiente. Dado que el intruso no conoce la clave secreta compartida, él o ella no podrá forjar paquetes.
- El router receptor debe calcular el *hash* MD5 sobre la información EIGRP y la clave secreta. Si el *hash* MD5 concuerda con la huella digital adjuntada al paquete, el paquete es considerado genuino y es aceptado para ser procesado. Los paquetes que no pasen la prueba de la huella MD5 son descartados.

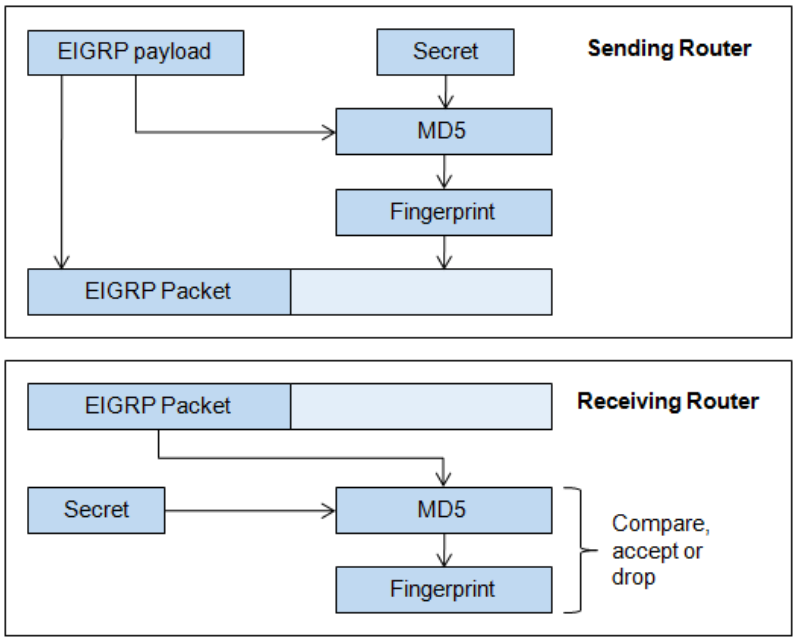

Figura 3.29: Autenticación MD5 en EIGRP

3.3 Redundancia en Redes EIGRP

Existen múltiples razones por las cuales se desearía incluir cierto índice de redundancia en una red, entre ellas se encuentran:

- Proveer enlaces entre la red en caso de que algún enlace o dispositivo falle.
- Proveer servicios óptimos de enrutamiento.
- Proveer balanceo de carga en las zonas altamente utilizadas.

Es muy común construir redes con routers *back-to-back* para crear redundancia. El protocolo de enrutamiento debe manejar a cada uno de los enlaces redundantes como una ruta de tránsito posible de forma que, cada enlace añada un set de rutas las cuales el protocolo debe tomar en cuenta a la hora de calcular rutas óptimas.

Las interfaces conectadas a enlaces de redundancia deben especificarse como interfaces pasivas de forma que no se formen relaciones con vecinos conectados a dichos enlaces. Al momento de diseñar una red EIGRP debe saberse que mientras más redundancia mayor complejidad se agrega a la red, los tiempos de convergencia aumentan al igual que los esfuerzos de administración y las posibilidades de problemas en la red.

3.4 Distancia Administrativa

La distancia administrativa (AD) es la confiabilidad (o preferencia) del origen de la ruta. EIGRP tiene una distancia administrativa predeterminada de 90 (ver Tabla 3.8) para las rutas internas y de 170 para las rutas externas, como rutas por defecto. Cuando se compara con otros protocolos IGP (*Interior Gateway Protocol*), EIGRP es el IGP que Cisco Systems prefiere porque cuenta con la distancia administrativa más baja [2].

Fuente del Prefijo	Distancia Administrativa
Conectada	0
Estática	1
EIGRP *summary route*	5
BGP Externo	20
EIGRP Interno	90
IGRP	100
OSPF	110
IS-IS	115
RIP	120
EIGRP Externo	170
BGP Interno	200

Tabla 3.8: Distancias administrativas

4. Marco Metodológico

Para el logro exitoso de los objetivos planteados en el Capítulo 1, es necesario definir un esquema o metodología de trabajo que permita el desarrollo rápido y eficiente de cada uno de los requerimientos de la aplicación. A continuación, se presenta la especificación de la metodología utilizada y otros detalles importantes que fueron tomados en cuenta para el desarrollo e implementación de la aplicación.

4.1 Adaptación de la Metodología de Desarrollo

La mayoría de las metodologías de desarrollo que existen plantean un esquema de trabajo que se divide en 4 fases: Análisis, Diseño, Codificación y Pruebas. Un conjunto de los modelos tradicionales proponen que dicho esquema se ejecute de forma lineal o secuencial para la implementación de la aplicación.

En la realidad, la adaptación de estos esquemas secuenciales a la práctica es poco frecuente, ya que durante el desarrollo de una aplicación, tienden a aparecer nuevos requerimientos durante la ejecución de cualquiera de las fases, ya sea porque el cliente tiene una nueva necesidad o porque al obtener resultados de la aplicación se decida incluir nuevos criterios, lo cual conlleva a la replanificación de las actividades, ejecutando nuevamente las 4 fases del método de desarrollo.

Por esta razón, se ha decidido trabajar con un método iterativo basado en el modelo de desarrollo ágil, el cual propone dividir el desarrollo de la aplicación por iteraciones y en cada iteración realizar las 4 fases del marco de trabajo tradicional. Es importante destacar que una iteración (bajo dicha metodología) puede ser vista de 2 formas: (1) como un período de tiempo (que varía entre 1 y 4 semanas) para el desarrollo de un grupo de requerimientos definidos durante la fase de Análisis de dicha iteración, o (2) como la implementación de un módulo de la aplicación.

La adaptación del esquema de trabajo escogido se basa en el modelo de desarrollo ágil, dividiendo los requerimientos de la aplicación por módulos, y desarrollando un módulo por iteración. Por tanto, en cada iteración se realizan las 4 fases del método tradicional. A continuación se describen los aspectos que se tomaron en cuenta en cada una de las fases de desarrollo.

4.1.1 Análisis y Planificación

Durante la fase de análisis se definen los requerimientos para cada iteración (módulo). Es importante destacar que por ser un proceso iterativo toda fase de análisis (excepto la primera iteración) es antecedida por una fase de pruebas del módulo que se estaba desarrollando.

Dichas pruebas pueden implicar el desarrollo de nuevos requerimientos al no obtener el resultado o comportamiento esperado, los cuales son analizados en una nueva iteración. En caso de que la fase de pruebas arroje los resultados esperados, se procede a la ejecución de una nueva iteración que incluirá el análisis de requerimientos del nuevo módulo a implementar.

4.1.2 Diseño

Durante la fase de diseño de cada iteración se crea la estructura lógica del módulo a desarrollar. Para ello, se definen las clases que van a interactuar, así como los métodos que deben ser incluidos dentro de cada una de las clases.

Esta fase debe ser documentada mediante el uso de diagramas de clases, los cuales pueden mostrar refinaciones de clases creadas en una iteración anterior o el diseño de nuevas clases del módulo a desarrollar.

4.1.3 Codificación

Una vez analizados los requerimientos y diseñada la solución para cubrir dichos requerimientos, se procede a codificar la misma. Es decir, se implementan las clases diseñadas y se desarrollan cada uno de los métodos definidos para las clases. Durante esta fase se documentan los aspectos de codificación más importantes por módulo.

4.1.4 Pruebas

Toda iteración culmina con la verificación de la solución creada para asegurar que se cumplan con los requerimientos planteados al inicio y determinar si a partir del desarrollo de la solución se desprenden nuevos requerimientos.

Al finalizar cada iteración se realizan pruebas de funcionamiento para validar que todos los datos arrojados por el módulo sean correctos, es decir, se verifica que cada módulo tenga el comportamiento esperado.

Luego, durante las pruebas de integración se analiza si la interfaz gráfica de usuario cubre todos los requerimientos necesarios para la recolección de datos y muestra de resultados y se llevan a cabo pruebas generales que garanticen que la aplicación funcione en conjunto y que cada uno de los módulos probados de forma independiente continúen funcionando de manera adecuada luego de la integración.

Los resultados de cada una de las pruebas descritas y los posibles nuevos requerimientos alimentan a la fase de análisis de la nueva iteración, la cual toma esta información unida a los requerimientos de otros módulos para determinar los requerimientos a cubrir en la próxima iteración.

4.2 Tecnologías a Utilizar

Tal y como se planteó en el Capítulo 1, para el desarrollo de la aplicación se utilizará Java como plataforma de desarrollo y un conjunto de paquetes compatibles con dicho lenguaje de programación que amplían las funcionalidades del mismo. Los paquetes o librerías a utilizar son: Jpcap, WinPcap, Libpcap, Devcon, JUNG y Log4j.

A continuación se describen brevemente cada una de las tecnologías mencionadas:

- Java: tecnología desarrollada por Sun Microsystems que ofrece la plataforma necesaria para la creación de interfaces gráficas (mediante el paquete Swing) y

todo el entorno de la aplicación. Para la creación del sistema será utilizada la versión 1.6.0_03 del JRE (*Java Runtime Environment*).

- Jpcap: paquete Java que permite la captura y envío de paquetes por la red, además del forjamiento de paquetes con opción de modificación de campos de la cabecera IP.
- WinPcap: herramienta estándar utilizada a nivel de enlace para proveer acceso a la red en ambientes Windows. La misma permite a las aplicaciones capturar y transmitir paquetes obviando la pila de protocolos.
- Libpcap: es una interfaz independiente del sistema para la captura de paquetes a nivel de interfaz de usuario. Esta librería es utilizada para ambientes Linux.
- Devcon: herramienta basada en línea de comandos que provee las funcionalidades del Administrador de Dispositivos (*Device Manager*). Gracias a esta herramienta, es posible habilitar, deshabilitar, reiniciar, actualizar, remover y hacer consultas sobre un dispositivo específico o sobre un grupo de ellos.
- JUNG: *Java Universal Network / Graph Framework* es una librería realizada en Java útil para el modelado, análisis y visualización de datos que se pueden representar en formas de gráficos.
- Log4j: es una librería de código abierto desarrollada en Java por *Apache Software Foundation* que permite elegir la salida y el nivel de granularidad de los mensajes o "logs" (*logging*) en tiempo de ejecución.

4.3 Prototipo General de Interfaz

Con el fin de mantener lineamientos estándar de interfaz gráfica entre los diferentes módulos de la aplicación, se diseñó un prototipo de interfaz general que establece dichos lineamientos y que se seguirán a lo largo del desarrollo de la aplicación para procurar la usabilidad y mantener la consistencia.

En la Figura 4.1 se muestra el prototipo de interfaz definido para la ventana principal, la cual será un JFrame con dimensiones de 700 píxeles de alto por 950 píxeles de ancho. En la barra de título, se mostrará el logo de la aplicación Easy-EIGRP.

Se ha decidido utilizar una estructura basada en pestañas para el acceso a las distintas herramientas de la aplicación con el objetivo de que el usuario pueda navegar fácilmente entre las diferentes funcionalidades de Easy-EIGRP y ofrecer así mayor flexibilidad al momento de ejecutarse varios módulos de forma simultánea.

Figura 4.1: Prototipo de interfaz ventana principal

Posteriormente, se diseñó el prototipo de interfaz para el panel general que se usará en cada una de las funcionalidades de la aplicación. Dicho prototipo es mostrado en la Figura 4.2, que propone que las dimensiones en píxeles del panel sean 655 de alto por 950 de ancho.

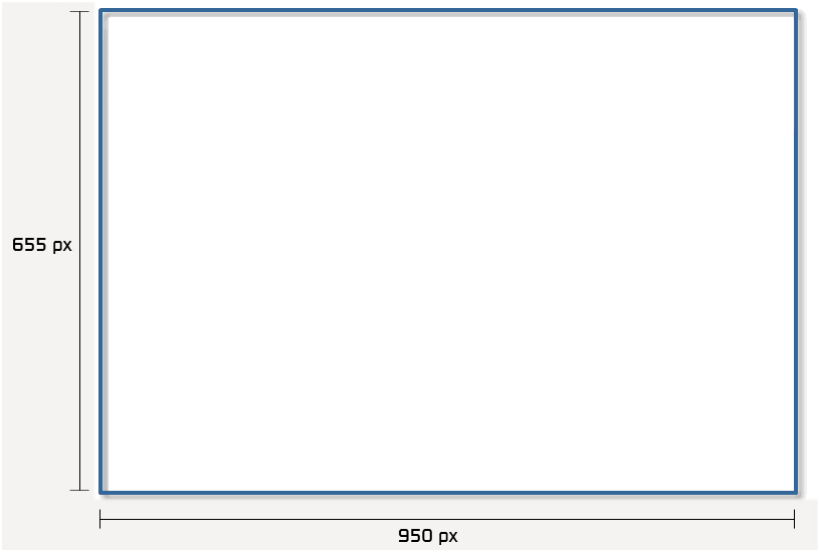

Figura 4.2: Prototipo de interfaz panel general

De ser necesario los paneles interiores contendrán un componente clave para facilitar el uso de las distintas herramientas, este componente es denominado JSplitPane, su función principal es optimizar el espacio del panel dividiéndolo en dos componentes que podrán ser cambiados de tamaño de forma interactiva por el usuario. Este componente puede ser encontrado tanto de forma horizontal como vertical o incluso ambas como se muestra en la Figura 4.3.

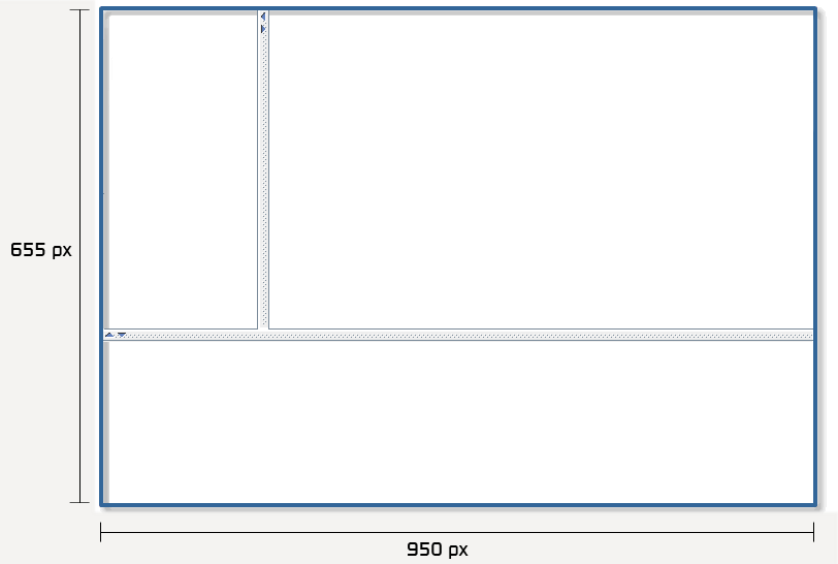

Figura 4.3: Prototipo de interfaz panel general con JSplitPane

5. Marco Aplicativo

En el Capítulo 4 se describió la metodología a utilizar para la implementación de la aplicación, la cual consiste en una metodología iterativa adaptada al modelo de desarrollo ágil, donde una iteración refleja el proceso de creación de un módulo de la aplicación.

Con el fin de simplificar y facilitar la comprensión de los resultados obtenidos durante cada iteración, los mismos son documentados agrupándolos por fases (Análisis, Diseño, Codificación y Pruebas) y especificando en cada una de ellas el detalle de todas las iteraciones. A continuación, se describe el proceso práctico que se siguió a lo largo del desarrollo de la aplicación.

5.1 Análisis General

Antes de iniciar el desarrollo de la aplicación, se llevó a cabo una fase de análisis general donde se determinó de manera global los principales requerimientos de la aplicación, los cuales deben cubrir los objetivos definidos en el Capítulo 1 para dar solución al problema.

A continuación, se muestra cómo se estructuró la lista de requerimientos de la aplicación:

- Definir una interfaz gráfica de usuario basada en el prototipo general de interfaz planteado en el Marco Metodológico que se fundamenta en los principios de usabilidad.
- Crear un entorno de configuración sobre los dispositivos presentes en el computador, incluyendo una vista en la interfaz para el mismo fin.
- Crear las estructuras, incluyendo paquetes y tablas, que el protocolo requiere.
- Crear vistas en la interfaz gráfica para observar el mapa parcial de la red.
- Crear vistas en la interfaz gráfica para manejar las tablas del protocolo.
- Implementar el protocolo de descubrimiento y recuperación de vecinos.
- Crear una bitácora o *logger* que incluya un depurador para observar las actividades llevadas a cabo por la aplicación.
- Implementar el protocolo RTP (*Realiable Transfer Protocol*).
- Implementar funcionalidades básicas del módulo DUAL (*Difussing Update Algorithm*).
- Implementar la máquina de estados finitos DUAL.
- Crear una vista que permita observar el estado de la máquina DUAL.

5.2 Desarrollo de la aplicación

El proceso de creación de Easy-EIGRP se dividió en 5 módulos, los cuales se listan a continuación:

1. Módulo *EIGRP Settings*.
2. Módulo *DUAL Finite State Machine*.
3. Módulo *Partial Network Map*.

4. Módulo *EIGRP Tables*.
5. Módulo *Logger*.

Cada módulo se encuentra fuertemente relacionado con el resto de ellos. Para definir la relación de los mismos se llevó a cabo un diagrama general de clases (Figura 5.1) el cual refleja dicha interacción.

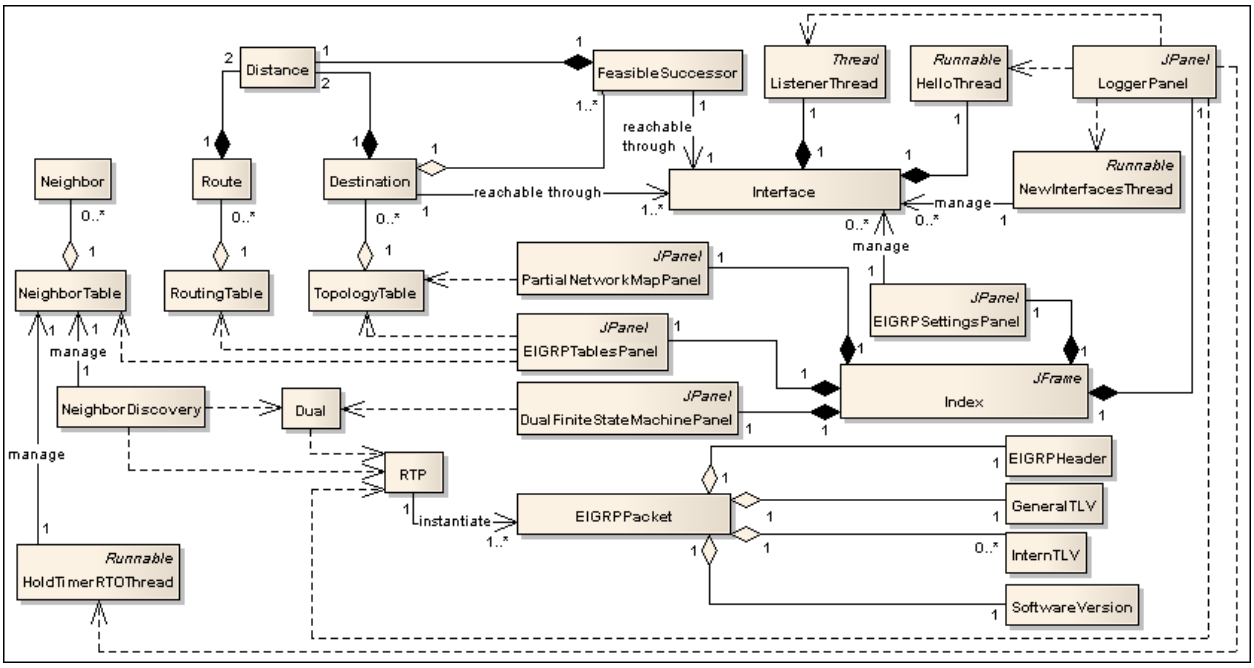

Figura 5.1: Diagrama de Clases Principales

A continuación se especifican cada una de las iteraciones correspondientes llevadas a cabo según la adaptación realizada del modelo de desarrollo ágil.

5.2.1 Iteración 1: Módulo *EIGRP Settings*

El módulo denominado *EIGRP Settings* se encuentra destinado al manejo y configuración de todos los aspectos tanto a nivel de interfaz de red como de atributos propios de EIGRP (*Enhanced Interior Gateway Routing Protocol*).

5.2.1.1 Fase de análisis

Para el módulo denominado *EIGRP Settings*, se planificó el desarrollo de una interfaz de usuario que ofreciera todas las posibilidades de configuración tanto de las interfaces de red del computador en el cual se ejecuta la aplicación como del protocolo EIGRP. Todas las configuraciones respectivas deben contar con las validaciones necesarias para controlar la entrada de datos correcta por parte del usuario.

Los atributos configurables fueron divididos en 2 niveles (nivel de interfaz de red y nivel de router), cada uno de ellos descritos a continuación.

1. Configuración a nivel de interfaz de red (*Network Interface Settings*): dichos cambios afectan únicamente a la interfaz en cuestión.

- Dirección IP de la interfaz de red: en este caso debe validarse que la dirección IP no se encuentre duplicada en la red (utilizando *Gratuitous ARP*), que no se encuentre solapada con otras direcciones y que tenga un formato válido para IPv4.
- Máscara de *subnet* de la interfaz de red: para las máscaras, las validaciones necesarias incluyen formato adecuado y coherencia con respecto a la dirección IP de la interfaz.
- Ancho de Banda (*Bandwidth*): expresado en Kbps. Se debe validar que los valores introducidos por el usuario tengan el formato válido y que además el valor se encuentre entre 1 y 10000000.
- Retardo (*Delay*): expresado en decenas de microsegundos. Se debe validar que los valores introducidos por el usuario tengan el formato válido y que además el valor introducido se encuentre entre 1 y 16777215.
- Estado EIGRP (*EIGRP Status*): con tres posibles valores: habilitado (*Enable*), pasivo (*Passive*) y deshabilitado (*Disable*). En caso de las interfaces tipo *loopback*, la opción de habilitar no debe encontrarse disponible.
- *Hold Time*: expresado en segundos y debe validarse que el valor introducido tenga el formato correcto y que se encuentre entre 1 y 65535. Además debe verificarse que el *hold time* sea extrictamente mayor que el *hello time*. Este atributo no debe ser configurable para las interfaces tipo *loopback*.
- *Hello Time*: expresado en segundos. Debe validarse que el formato sea el adecuado, que el valor introducido se encuentre entre 1 y 65535 y que además el *hello time* sea extrictamente menor que el *hold time*. Este atributo no debe ser configurable para las interfaces tipo *loopback*.
- *Summarization* Manual (*Manual Summarization*): con posibilidades de ser habilidado (*Enable*) o desahbilitado (*Disable*). En caso de ser habilitado el usuario debe tener la posibilidad de ingresar los datos necesarios (dirección de *subnet* y máscara respectiva) para llevar a cabo el proceso de *summarization*. Las validaciones deben ser semejantes a las mencionadas anteriormente tanto para las direcciones IP como para las máscaras de *subnet* (se encuentran en formato *CIDR*), tomando en cuenta además que en este caso la dirección IP debe corresponder exactamente a la máscara introducida, es decir, al realizar una operación lógica *and* bit a bit entre la dirección IP y la máscara, el resultado debe ser la dirección IP. Este atributo no debe encontrarse habilitado para las interfaces tipo *loopback*.

2. Configuración a nivel de router (*Router Settings*): afecta todas las interfaces de red.
    - Número de Sistema Autónomo (*AS*): debe verificarse que sea un número y que el valor introducido se encuentre entre 1 y 65535.
    - Máximo de Retransmisiones Permitidas (*Max Retransmitions Allowed*).

- K1: se debe verificar que el formato sea adecuado y que el valor introducido este entre 0 y 255 y que además no sea 0 simultáneamente con K3.
- K3: se debe verificar que el formato sea adecuado y que el valor introducido este entre 0 y 255 y que además no sea 0 simultáneamente con K1.
- *Summarization* Automática (*Auto Summarization*).
- Varianza (*Variance*): se debe verificar que sea un número entre 1 y 128.
- Temporizador de rutas SIA (*SIA Timer*): se mide en minutos, debe ser un número comprendido entre 1 y 65535.

5.2.1.2 Fase de Diseño

En la fase de diseño del módulo *EIGRP Settings* se definieron varias clases las cuales permitieron la posterior codificación del mismo. En la Figura 5.2 se muestra de forma general las clases más importantes con sus respectivas interacciones, incluyendo sólo los atributos y métodos más importantes.

La clase *EIGRPSettingsPanel* se encarga del manejo tanto visual como funcional de las interfaces de red del computador y de los atributos configurables de EIGRP.

En la Figura 5.2 se puede apreciar las interacciones de mayor relevancia de la clase *EIGRPSettingsPanel* las cuales involucran a la clase *Interface* y a la clase *Index*.

La clase *Interface* se encarga de generar una estructura de datos la cual representa una interfaz física o loopback EIGRP.

Tanto el estado de una interfaz como los atributos de la misma pueden ser alterados por la clase *NewInterfacesThread*. Dicha clase se encuentra alerta a cualquier cambio realizado tanto por el usuario, los cuales son capturados vía GUI (*Graphic User Interface*), como por el entorno ajeno al usuario. En caso de que exista alguna alteración que no haya sido generada a través del GUI, *NewInterfacesThread* modificará la estructura de datos tipo *Interface* correspondiente quien a su vez informará del cambio a *EIGRPSettingsPanel*.

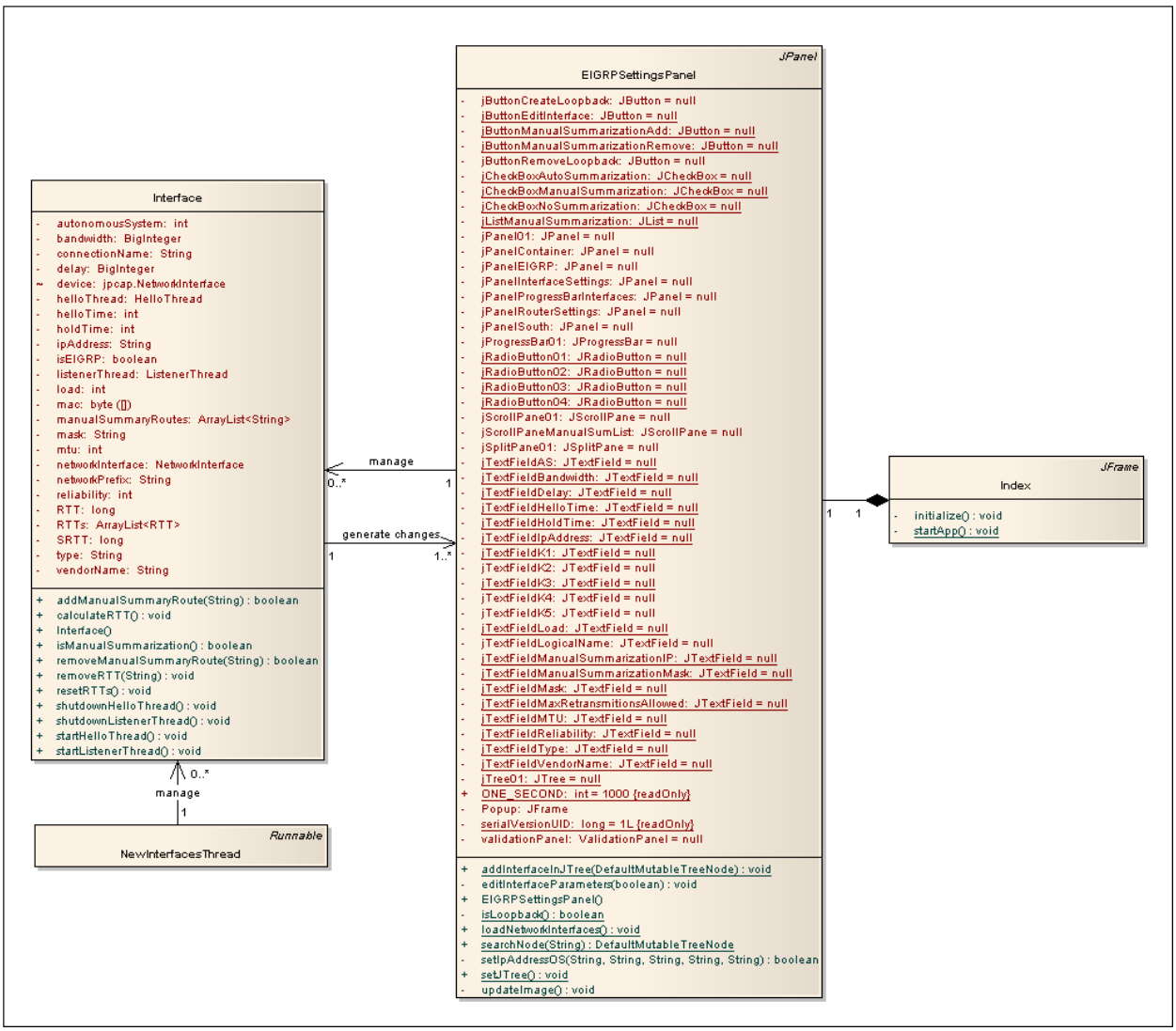

Figura 5.2: Diagrama de Clases General del Módulo *EIGRP Settings*

5.2.1.3 Fase de Codificación

Para la creación del módulo *EIGRP Settings* se definió una interfaz gráfica que permitiera al usuario introducir todos los campos necesarios para la configuración del protocolo. Adicionalmente, se desarrollaron métodos para validar que los valores introducidos por el usuario fueran correctos.

En la Figura 5.3 se muestra un fragmento del código desarrollado para capturar cambios generados por el usuario desde el GUI y llevar a cabo las tareas necesarias. Más específicamente se muestran las acciones que se realizan al capturar un cambio de *No-Summarization* a *AutoSummarization*.

```
private void editInterfaceParameters(boolean isLoopback, boolean setIpMask){


    /*
     * SET SUMMARIZATION STATES
     */
    /*
     * AUTO SUMMARIZATION
     */
    if(jCheckBoxAutoSummarization.isSelected() && Singleton.isAutoSum()==false){
        setManualSumValidationActive(false);
        clearManualSumValidations();


        ((DefaultListModel) jListManualSummarization.getModel()).clear();
        for(int x=0;x<Singleton.getMyInterfaces().size();x++){
            if(Singleton.getMyInterfaces().get(x).isManualSum()){
                Singleton.getMyInterfaces().get(x).manageManualSummaryRoutes(new ArrayList<String>());
            }

            Singleton.getMyInterfaces().get(x).setManualSum(false);
        }

        Singleton.getDual().setSummarization(Utils.getInterfaceByLogicalName(jTextFieldLogicalName.getText()), true);
        Singleton.setAutoSum(true);
        jRadioButtonManualSummarization.setEnabled(false);

    /*
     * NO AUTO SUMMARIZATION
     */
    }else if(!jCheckBoxAutoSummarization.isSelected() && (Singleton.isAutoSum() || Utils.getInterfaceByLogicalName(jTextFieldLogicalName.getText()).isManualSum()
            || (!Singleton.isAutoSum() || !Utils.getInterfaceByLogicalName(jTextFieldLogicalName.getText()).isManualSum()))){
        jRadioButtonManualSummarization.setEnabled(true);
        setManualSumValidationActive(true);


        /*
         * MANUAL SUMMARIZATION ENABLED
         */
        if(jRadioButtonManualSummarization.isSelected()){

            /*
             * FIRST TIME MANUAL SUM
             */
            if(!Utils.getInterfaceByLogicalName(jTextFieldLogicalName.getText()).isManualSum()){
                jCheckBoxAutoSummarization.setEnabled(false);
                setManualSumValidationActive(true);
                Utils.getInterfaceByLogicalName(jTextFieldLogicalName.getText()).setManualSum(true);

                jTextFieldManualSummarizationIP.setText("");
```

Figura 5.3: Segmento de código para llevar a cabo cambios generados por el usuario en las interfaces

5.2.1.4 Fase de Pruebas

Las pruebas unitarias de funcionamiento realizadas para este módulo, consisten en verificar que cada una de las opciones de los menús provistos en la barra de tareas superior sean coherentes con las acciones llevadas a cabo. Adicionalmente, se valida que los atajos para usuarios expertos conduzcan a la pestaña indicada.

También se llevaron a cabo pruebas de casos bordes en las entradas de los campos pertenecientes al panel de *EIGRP Settings*, de esta forma se asegura que el usuario no pueda introducir valores incoherentes o erróneos y que por ende el funcionamiento de la aplicación sea correcto.

En la Figura 5.4 se puede apreciar el panel correspondiente al módulo *EIGRP Settings*. En la parte superior puede observarse la barra de tareas la cual cuenta con 4 menús (los cuales incluyen sus respectivos atajos o *shortcuts*) la cual se encuentra disponible desde cualquier panel o módulo:

1. *File*: provee opciones de:
    - *Open:* referente a configuración XML *(Extensible Markup Language)*[8].
    - *Save:* referente a configuración XML.

[8] http://www.w3.org/XML/

- *Save As:* referente a configuración XML.
- *Quit.*

2. *Edit*: despliega un sub-menú que contiene:
   - *Cut:* referente a edición de texto plano.
   - *Copy:* referente a edición de texto plano.
   - *Paste:* referente a edición de texto plano.

3. *View*: permite configurar la cantidad de pestañas o *tabs* accesibles en un instante de tiempo.
4. *Help*: proporciona la ayuda necesaria al usuario para el manejo de la aplicación.

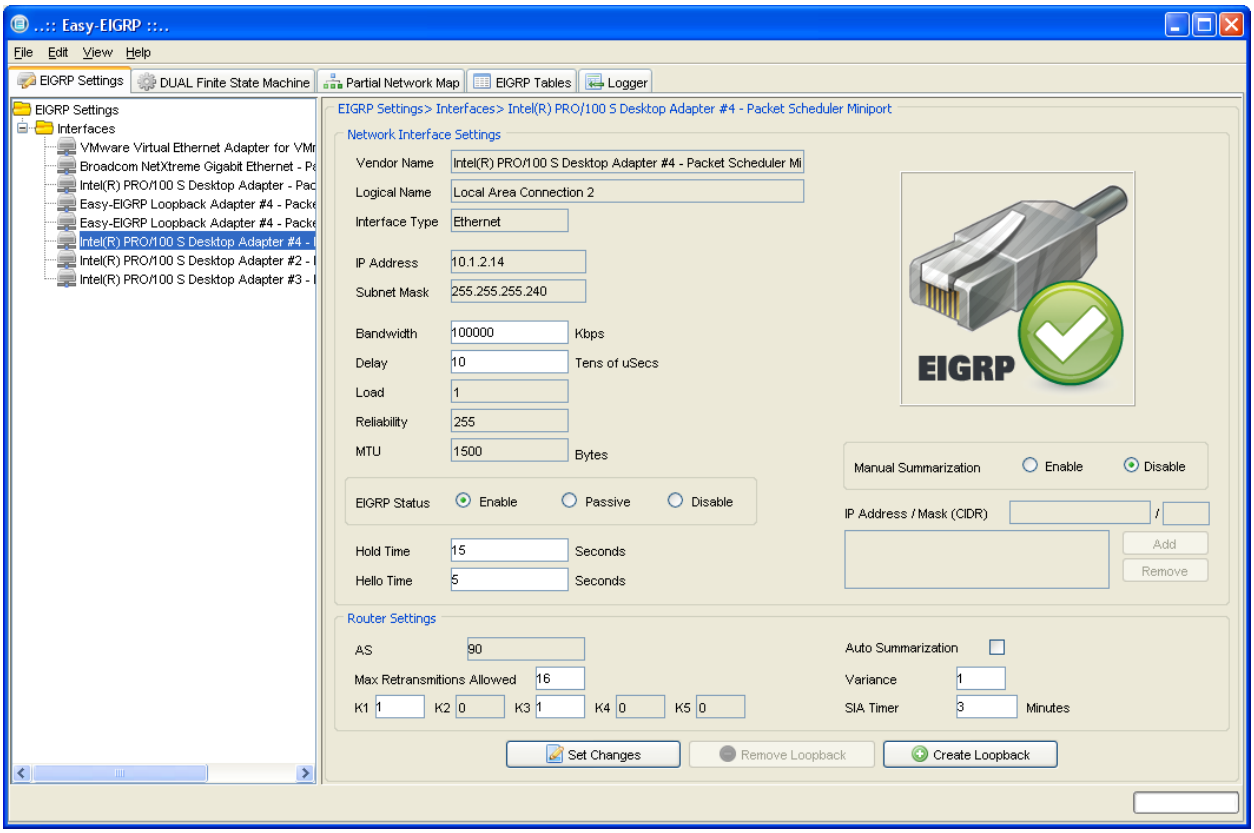

Figura 5.4: Panel *EIGRP Settings*

5.2.2 Iteración 2: Módulo DUAL Finite State Machine

El módulo que se presenta a continuación es denominado *DUAL Finite State Machine*, tiene la finalidad de mostrar de manera sencilla y dinámica el proceso de *diffusing computations*, el cual es discutido en la Sección 3.1.4 del Capítulo 3.

5.2.2.1 Fase de análisis

El objetivo principal en cuanto a la interfaz gráfica de este módulo es proveer las herramientas necesarias al usuario para observar cada paso del proceso de *diffusing computations*, mostrar en tiempo real los cambios en la tabla *Reply-Status*, así como

también actualizar la máquina de estados finitos dependiendo de los *input events* que se presenten. Este panel debe tener 5 secciones principales:

1. Lista de prefijos: en la cual se muestra cada prefijo nuevo al momento de ser aprendido.
2. Imagen de la máquina de estados finitos: se refiere a una imagen dinámica que se actualiza dependiendo de los cambios de estados para el prefijo seleccionado de la lista.
3. *Logger* del panel: muestra cada evento que se genera en el panel, desde los *queries* enviados, *replies* recibidos hasta los *input events* que causan los cambios de estado en la imagen.
4. Tabla *Reply-Status*: es una tabla dinámica que se actualiza cada segundo mostrando los *queries* enviados a los vecinos y *replies* de los mismos, de igual manera muestra al usuario los estados de cada vecino en caso de entrar en SIA (*Stuck In Active*).
5. Controles del panel: esta sección otorga al usuario ciertas funcionalidades para poder navegar a través de todos los pasos del proceso, ya que puede terminar en pocos segundos, permitirá regresar a estados anteriores del proceso y poder recrear el proceso de *diffusing computations* nuevamente.

5.2.2.2 Fase de Diseño

El diseño del módulo *DUAL Finite State Machine* es sencillo y tiene el único objetivo de permitir al usuario aprender sobre la máquina de estados finitos de EIGRP de forma dinámica e interactiva. En la Figura 5.5 se puede observar el resultado final de este módulo.

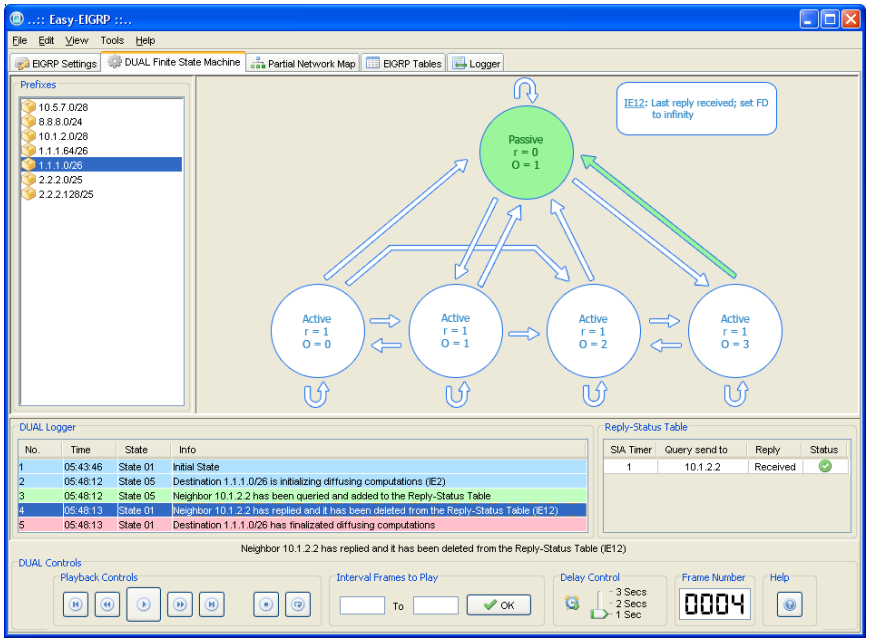

Figura 5.5: Interfaz gráfica del Panel *DUAL Finite State Machine*

Durante esta fase se crearon una cierta cantidad de clases tanto para la creación de estructuras de almacenamiento (*InputEvent* y *Prefix*) como para la interfaz gráfica, cada una de ellas vistas en el diagrama de clases general de la Figura 5.6.

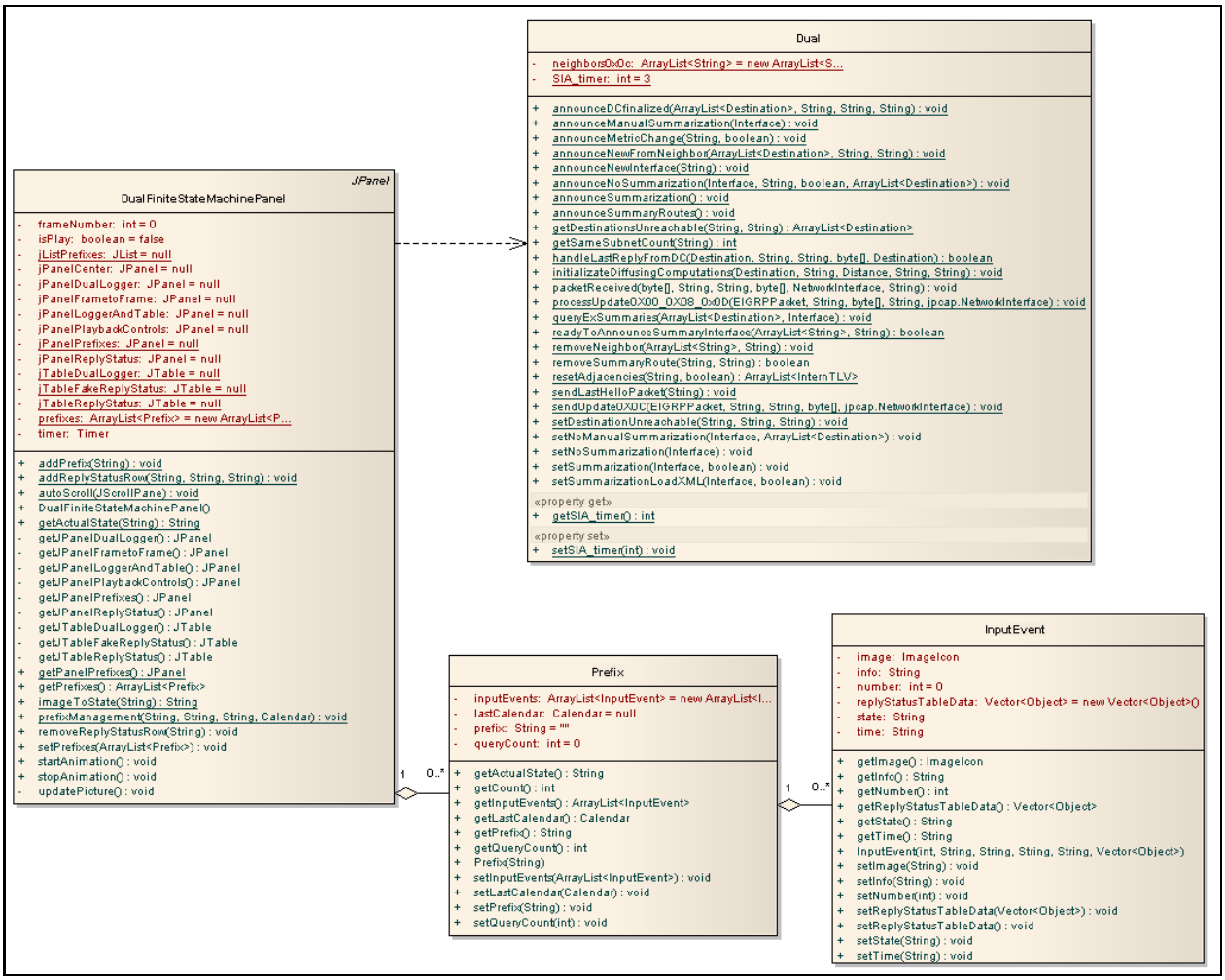

Figura 5.6: Diagrama de Clases General del Módulo *DUAL Finite State Machine*

El proceso de *diffusing computations* se produce para un prefijo determinado, por lo tanto se debe almacenar cada evento producido en una estructura para dicho prefijo, como se observa en la clase *DualFiniteStateMachine en la que* se cuenta con un *ArrayList* de tipo *Prefix* llamado *prefixes* el cual a su vez posee otro *ArrayList* de tipo *InputEvent* para almacenar los distintos eventos que conformen el *diffusing computation.* Una vez llenada esta estructura se tiene de manera permanente el o los procesos completos de *diffusing computations* que para efectos de enseñanza, el usuario podrá repetir a su disposición cuantas veces lo desee.

5.2.2.3 Fase de Codificación

Para la codificación de este módulo, al igual que para los otros, se siguió con los lineamientos del patrón de diseño *Singleton* para garantizar una única instancia de este panel. Además, se definió una interfaz gráfica que permitiera al usuario observar cada paso en el proceso de *diffusing computations* para un prefijo seleccionado.

Similar a los demás paneles, *DUAL Finite State Machine* procura ser totalmente modular, ya que comparte sub paneles para cada sección, específicamente para desarrollo de la imagen de la máquina de estados finitos se crearon un total de 39 *frames* que definen cada estado y la transición entre ellos.

Junto con estas imágenes se creó una pequeña estructura para relacionar unívocamente una imagen con un *input event* y de esta forma generar la animación correspondiente dependiendo del proceso de *diffusing computation* que se haya generado y la tabla *Reply-Status*.

Como podemos observar en la Figura 5.7, dependiendo del prefijo, se crea un nuevo objeto del tipo *InputEvent* que estará identificado con un número único ascendente y además estará compuesto por el tiempo del evento, un identificador para el próximo estado esperado, un *String* información adicional y la imagen correspondiente.

Los eventos que son almacenados en el *ArrayList* del prefijo que ocasionó el *diffusing computation,* son mostrados en el *logger* del módulo para generar la animación de los estados y la transición entre ellos.

```
public static void prefixManagement(String prefix, String nextState, String message, Calendar calendar){
    for(int i=0; i<prefixes.size(); i++){
        if(prefixes.get(i).getPrefix().equalsIgnoreCase(prefix)){

            /**
             *  New Input Event
             **/
            InputEvent inputEvent = new InputEvent();

            //Set count of inputEvents and increase it
            prefixes.get(i).setCount(prefixes.get(i).getCount()+1);
            inputEvent.setNumber(prefixes.get(i).getCount());

            //Input Event number
            int number = inputEvent.getNumber();

            //Input Event time
            String time = sdf.format(new Date());
            inputEvent.setTime(time);

            /*
             * if nextState == ""       then stay in the same state
             * if nextState == "XXX"    then stay in the same state with the arrow
             */

            if(nextState.equalsIgnoreCase("") || nextState.equalsIgnoreCase("XXX")){
                inputEvent.setState(prefixes.get(i).getInputEvents().get(prefixes.get(i).getInputEvents().size()-1).getState());
                inputEvent.setImage(getNewImage(prefix, nextState));
                inputEvent.setImageName(getNewImage(prefix, nextState));
            }else{
                inputEvent.setState(nextState);
                inputEvent.setImage(getNewImage(prefix, nextState));
                inputEvent.setImageName(getNewImage(prefix, nextState));
            }
```

Figura 5.7: Segmento de código para el manejo de los prefijos

5.2.2.4 Fase de Pruebas

Para esta fase se diseñaron varios escenarios de pruebas para generar procesos de *diffusing computation*. Básicamente los *diffusing computations* se llevan a cabo siempre y cuando ocurra que:

- Se pierde la ruta actual y no se encuentra una ruta alterna.

- Se pierde la ruta actual y la nueva encontrada incluye al *successor* que causó un aumento de métrica.
- Se pierde la ruta actual y la nueva encontrada no incluye a un *feasible successor*.

Con el conocimiento anterior, se llevaron a cabo pruebas en las que las estructuras debían ser llenadas y así generar la animación correspondiente en tiempo real.

5.2.3 Iteración 3: Módulo *Partial Network Map*

Este módulo proporciona una vista gráfica, coherente y actualizada en tiempo real de la red parcial, obtenida a partir de las adyacencias y el intercambio de información con otros routers y/o equipos.

Al igual que el módulo anterior (ver Iteración 2: Módulo DUAL Finite State Machine), este módulo también tiene un *logger* que describe todos los cambios o actualizaciones que se han producido en la red.

El *Partial Network Map* es especialmente adecuado tanto para la enseñanza como para el aprendizaje de la métrica de EIGRP, ya que toda la información pertinente es presentada gráficamente.

5.2.3.1 Fase de análisis

El desarrollo de la interfaz gráfica de este módulo debe ser completamente interactivo y amigable con un eficiente uso del espacio para la buena visión del mapa parcial de la red.

Este módulo debe constar con una serie de herramientas que le permitan al usuario poder cambiar la posición de cada nodo, exportar el mapa parcial de la red en formato PNG o JPEG y poder alejar o acercar (*Zoom Out* o *Zoom In*) el mismo.

5.2.3.2 Fase de Diseño

El diseño del módulo *Partial Network Map* está basado en el uso de *JUNG* [28] (*Java Universal Network Graph*), un paquete que proporciona un lenguaje común y extensible para el modelado, análisis y visualización de datos que se pueden representar en un gráfico.

Por otro lado, este módulo debe tener una estrecha relación con cada una de las tablas de EIGRP para mantener una coherencia en la aplicación a la hora de detectar nuevas adyacencias o pérdida de las mismas, descubrir nuevos vecinos, actualizar métricas, entre otros eventos, para ello se desarrollaron varias clases de las cuales las más importantes se pueden observar en la Figura 5.8.

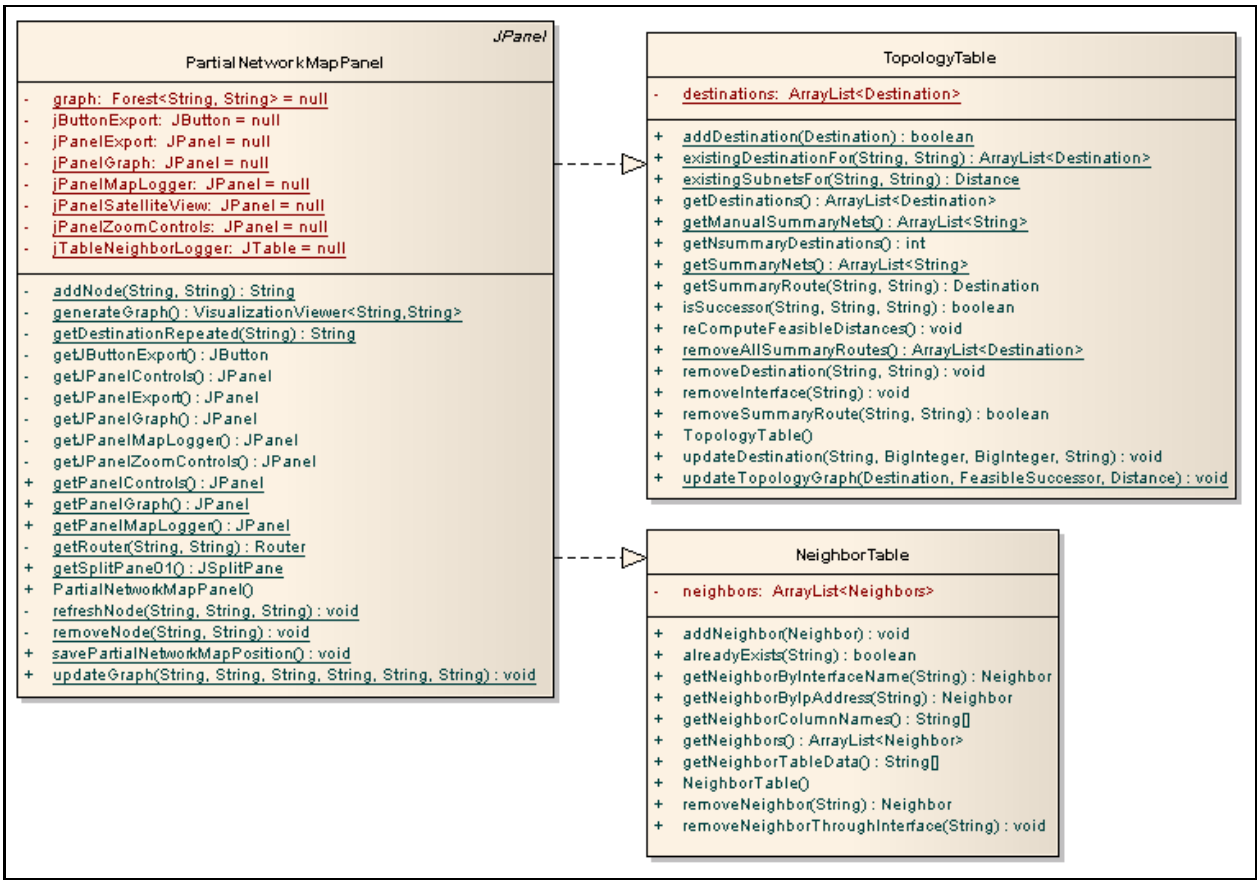

Figura 5.8: Diagrama de Clases General del Módulo *Partial Network Map*

Para este panel se emplean imágenes estándares en cuanto a redes se refiere, tales como routers y nubes, asociándolas a los nodos del mapa parcial de la red para representar routers vecinos y *subnets* asociadas a estos. Estas imágenes pueden observarse en la Figura 5.9. Así mismo, para el *logger* que posee este módulo al igual que los demás se emplean colores como el azul, verde y rojo para diferenciar cada tipo de mensaje mostrado.

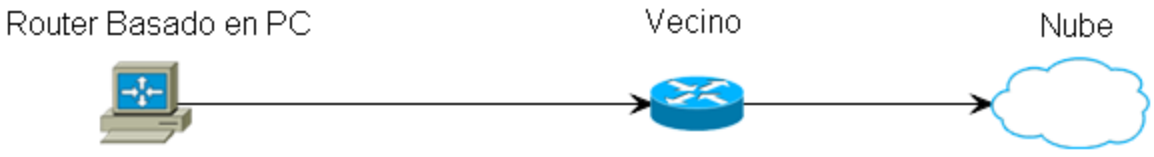

Figura 5.9: Imágenes para el Mapa Parcial de la Red

5.2.3.3 Fase de Codificación

Esta fase consta del desarrollo de cada clase definida en la fase de diseño, y al igual que los demás módulos, en *Partial Network Map* se emplea el patrón de diseño *Singleton* para garantizar una sola instancia del panel.

Como se puede observar en la Figura 5.10 dependiendo de la operación a realizar se agregan, eliminan o actualizan nodos en el mapa, esto permite una gran flexibilidad ya

que el manejo de vecinos y *subnets* se encuentra distribuido en los distintos módulos, por lo tanto esta función será llamada en cada punto del proceso de creación o destrucción de vecindades. De igual forma esta función agrega al *logger* correspondiente la información necesaria para que el usuario esté al tanto de lo sucedido en la topología.

Ya que la comunicación de este módulo es directa con la tabla de vecinos, constantemente se está recorriendo para actualizar el mapa y detectar la llegada de nuevos vecinos así como la perdida de los mismos.

En cuanto a la interfaz gráfica, este panel se divide en 3 secciones principales:

- *Network Partial Graph*: Se refiere al panel donde se dibujara el mapa parcial de la red.
- *Logger*: Es la sección que administra cada uno de los mensajes actualizando constantemente un *JTable* para el almacenamiento de los mismos.
- *Graph Control*: Esta sección posee distintas herramientas que amplían las posibilidades del usuario para navegar y manejar el mapa parcial de la red, entre ellas tenemos: acercamiento y alejamiento del mapa, vista satélite del mismo, manejo *drag and drop* del mapa completo o de cada uno de los nodos que lo conforman y la posibilidad de exportar el mapa en formato *PNG* (*Portable Network Graphics*) o *JPEG* (*Joint Photographic Experts Group*).

```
public static void updateGraph(String info, String operation, String type, String origin, String oldNode, String newNode){

    if(operation.equalsIgnoreCase("add")){
        addNode(type, origin);
    }else if(operation.equalsIgnoreCase("remove")){
        //this time only newNode is neigh.getInterfaceName()
        if(type.equalsIgnoreCase("neighbor"))
            saveNeighborAndCloudXY(oldNode, newNode);
        removeNode(type, oldNode);
    }else if(operation.equalsIgnoreCase("refresh")){ //refresh
        refreshNode(type, origin, oldNode);
    }

    addIcons();

    ((JPanel)jSplitPane01.getLeftComponent()).repaint();
    satellite.repaint();

    Date date = new Date();
    SimpleDateFormat sdf = new SimpleDateFormat("hh:mm:ss");

    jTableNeighborLogger.requestFocus();

    ((javax.swing.table.DefaultTableModel)jTableNeighborLogger.getModel()).insertRow(jTableNeighborLogger.getModel().getRowCount(),
            new String[]{""+Singleton.getMyPartialMapLoggerNo(), sdf.format(date) , info});
    jTableNeighborLogger.getSelectionModel().setSelectionInterval(jTableNeighborLogger.getRowCount()-1,
            jTableNeighborLogger.getRowCount());
    autoScroll(jScrollPane01);

}
```

Figura 5.10: Segmento de código para la actualización del *Partial Network Map.*

El resultado final de este módulo junto con cada una de estas secciones puede observarse en la Figura 5.11.

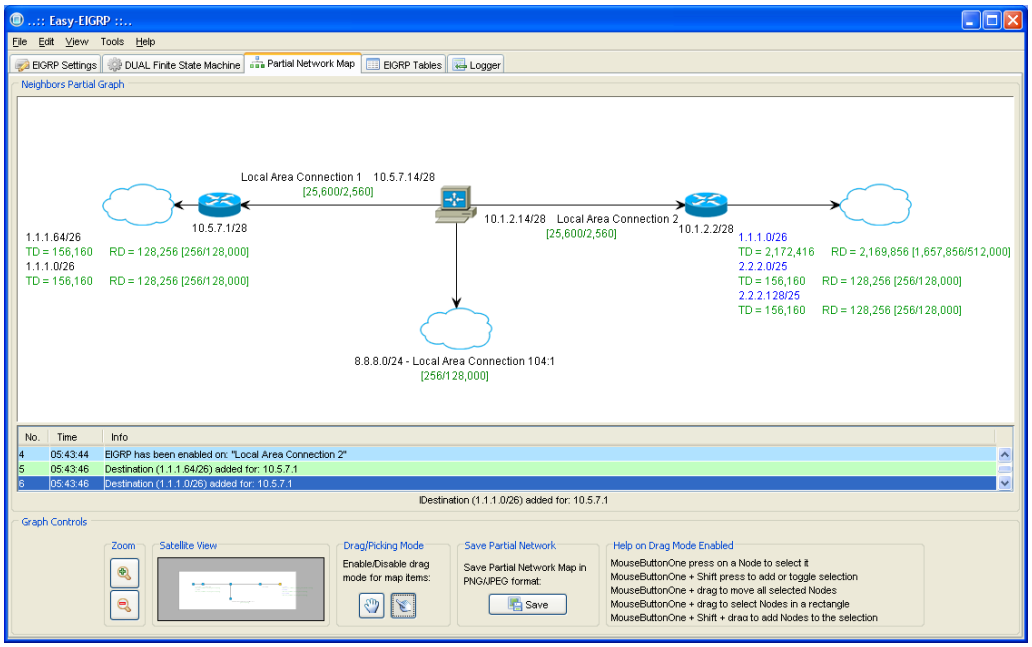

Figura 5.11: Panel *Partial Network Map*

5.2.3.4 Fase de Pruebas

Varios casos de prueba fueron empleados para esta fase, los cuales serán explicados en la Sección 5.3. Básicamente la prueba gira en torno a la creación de vecindades con otros routers y a la modificación de valores de los mismos como *bandwidth* y *delay*, para crear una red visible en la aplicación en la que se reflejen los cambios y actualizaciones, para de esta manera observar y comparar los resultados con las distintas tablas de EIGRP y verificar posibles errores o inconsistencias.

5.2.4 Iteración 4: Módulo *EIGRP Tables*

El módulo que se presenta a continuación es denominado *EIGRP Tables*, el cual, como su nombre lo indica, es el encargado de gestionar las estructuras de datos fundamentales para el funcionamiento de EIGRP. Las estructuras son representadas como tablas y se cuenta con cuatro de ellas, las cuales se describen en el Capítulo 3.

5.2.4.1 Fase de análisis

El desarrollo de la interfaz gráfica del módulo *EIGRP Tables* debe ser un reflejo exacto de las estructuras que maneja Easy-EIGRP, es decir, todas las modificaciones realizadas a nivel de protocolo deben ser reflejadas en tiempo real en la interfaz de usuario.

Las cuatro tablas a desarrollar son:

1. *Neighbor Table*: Se refiere a la tabla de vecinos y debe contar con nueve columnas las cuales describirán propiedades de los vecinos conocidos a través del protocolo.
2. *IP Routing Table*: La tabla de enrutamiento debe constar de cuatro columnas expandiendo el conocimiento de las rutas las cuales maneja Easy-EIGRP en un

instante dado. Los cambios realizados en esta tabla deben reflejarse en la tabla de enrutamiento del computador.

3. *EIGRP Topology Table*: La tabla topológica, según las especificaciones de EIGRP, debe contar con tres columnas las cuales identifiquen los posibles sucesores para un prefijo dado.
4. *Complete EIGRP Topology Table*: Dicha tabla debe ser una extensión de la *EIGRP Topology Table*, lo que quiere decir que ésta debe proveer mayor información sobre las distintas formas de llegar a un prefijo, por ejemplo, los vecinos que no son tomados como *feasible successors* en un instante de tiempo específico.

5.2.4.2 Fase de Diseño

El diseño del módulo *EIGRP Tables* constó del desarrollo de un cierto número de clases, las más relevantes se pueden apreciar en la Figura 5.12 y en la Figura 5.13, las cuales representan un único diagrama de clases, sin embargo fueron divididos para poder ser mejor visualizados.

En ambas figuras se puede observar que existe una clase *EIGRPTablesPanel*, la cual se encarga de la visualización gráfica por medio del GUI y representa las estructuras de datos desarrolladas y almacenadas por las clases *NeighborTable, RoutingTable* y *TopologyTable.*

La tabla de vecinos, representada por la clase *NeighborTable* se encuentra directamente manejada por la clase *NeighborDiscovery* (la cual se corresponde a uno de los módulos dependientes de EIGRP), quien administra el descubrimiento de nuevos vecinos gracias al trabajo del DUAL y del RTP (descritos en el capítulo anterior). *NeighborTable* se encuentra compuesta por una serie de objetos tipo *Neighbor*, los cuales poseen la información requerida por el protocolo para cada vecino.

El *ListenerThread* es la clase tipo *Thread* o hilo, que se encarga de administrar la recepción y envío de paquetes EIGRP y por ende, es quien recibirá los paquetes *hello* de un nuevo vecino, los cuales serán procesados por *NeighborDiscovery* y la clase DUAL.

La clase RTP es quien crea los paquetes EIGRP que deben ser intercambiados por la red. Dichos paquetes se encuentran generalizados en la clase *EIGRPPacket*, la cual posee un atributo para cada sección de los paquetes del protocolo. Estos objetos son del tipo *EIGRPHeader*, *GeneralTLV*, *InternTLV* y *SoftwareVersion*.

La tabla de enrutamiento o *RoutingTable* se encuentra compuesta por cero o más objetos *Route*, los cuales representan rutas hacia prefijos específicos y entre sus atributos se encuentran dos objetos tipo *Distance*, los cuales se refieren a la *Reported Distance* y a la *Feasible Distance* que cada ruta debe poseer.

La clase *TopologyTable* representa la tabla topológica EIGRP y está compuesta por objetos *Destination* los cuales están compuestos a su vez por uno o más *FeasibleSuccessor*, quienes según el valor de su atributo *TopologyTableAll* serán

reflejados en la *Topology Table* o en la *Complete Topology Table*. Al igual que los objetos *Route*, los *FeassibleSuccessor* se encuentran compuestos por dos distancias, una correspondiente a la *Reported Distance* y otra a la *Feasible Distance*.

En esta fase se decidió utilizar colores en las tablas para facilitar la visualización y entendimiento de las mismas, además de proporcionar la opción de auto refrescamiento de las tablas o de refrescamiento manual dependiendo de la finalidad del usuario.

5.2.4.1 Fase de Codificación

Para esta fase, se llevó a cabo el desarrollo de las clases anteriormente expuestas y se creó su visualización en el GUI.

Para la codificación de este módulo se procuró mantener una similitud estrecha con la teoría estudiada previamente sobre EIGRP, de esta forma se facilitó el proceso y se aseguró que las modificaciones futuras no sean complejas, gracias al nivel de abstracción utilizado. En la Figura 5.14 puede apreciarse un fragmento de código el cual corresponde a la creación de la tabla topológica y a la obtención de un prefijo dado su dirección IP. Además, se muestran algunas funciones minimizadas en la parte posterior.

5.2.4.2 Fase de Pruebas

La fase de pruebas en este caso no requirió de entradas por el usuario, simplemente se crearon un gran número de escenarios de prueba en donde los prefijos debían ser modificados, eliminados o agregados y, comparando con tablas proporcionadas por routers Cisco (bajo los mismos escenarios), se observaron errores los cuales fueron rectificados logrando así el correcto funcionamiento del módulo.

En la Figura 5.15 puede apreciarse la selección de las diferentes tablas mencionadas con anterioridad, así como la opción de auto refrescamiento y el botón de refrescamiento manual. Junto con la selección de las tablas se encuentran a modo de leyenda los códigos posibles encontrados en la tabla actualmente seleccionada.

En la Figura 5.16 se muestra la vista de la tabla de vecinos, adicionalmente puede apreciarse la tabla de enrutamiento en la Figura 5.17, mientras que en las Figura 5.18 y en la Figura 5.19 pueden observarse la tabla topológica y la tabla topológica completa respectivamente. En las dos últimas imágenes puede observarse la diferencia entre ambas tablas, por ejemplo, en la Figura 5.19 se puede observar una entrada para un *No Feasible Successor* para el prefijo 1.1.1.0/25, lo que no puede verse en la Figura 5.18 debido a la naturaleza de esta tabla.

Cabe destacar que tanto en la *IP Routing Table* como en las *Topology Table*, las rutas *summarized* son coloreadas amarillas, mientras que las demás rutas son agrupadas por colores verde o azul de forma intercalada.

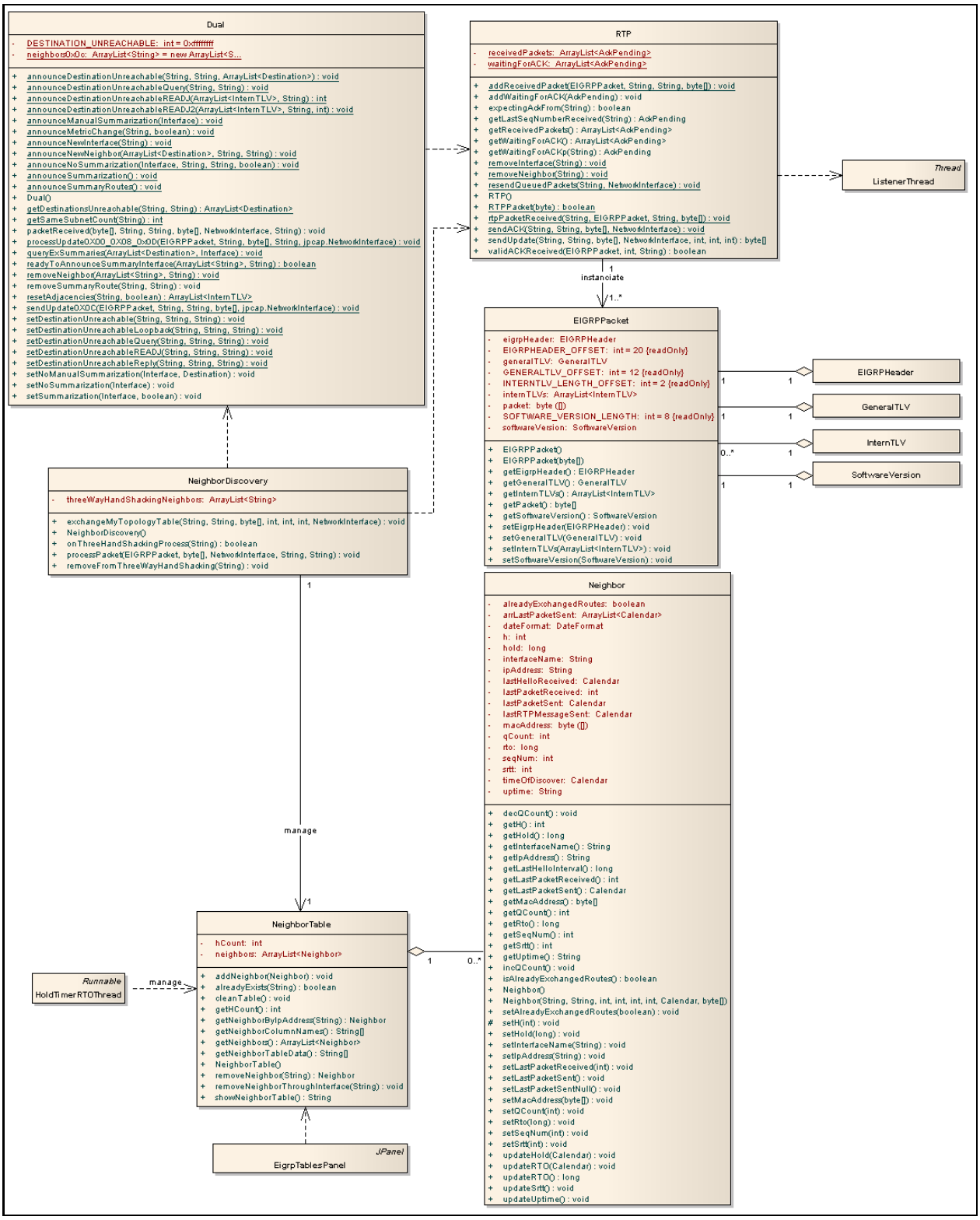

Figura 5.12: Diagrama de Clases General del Módulo *EIGRP Tables* (Parte I)

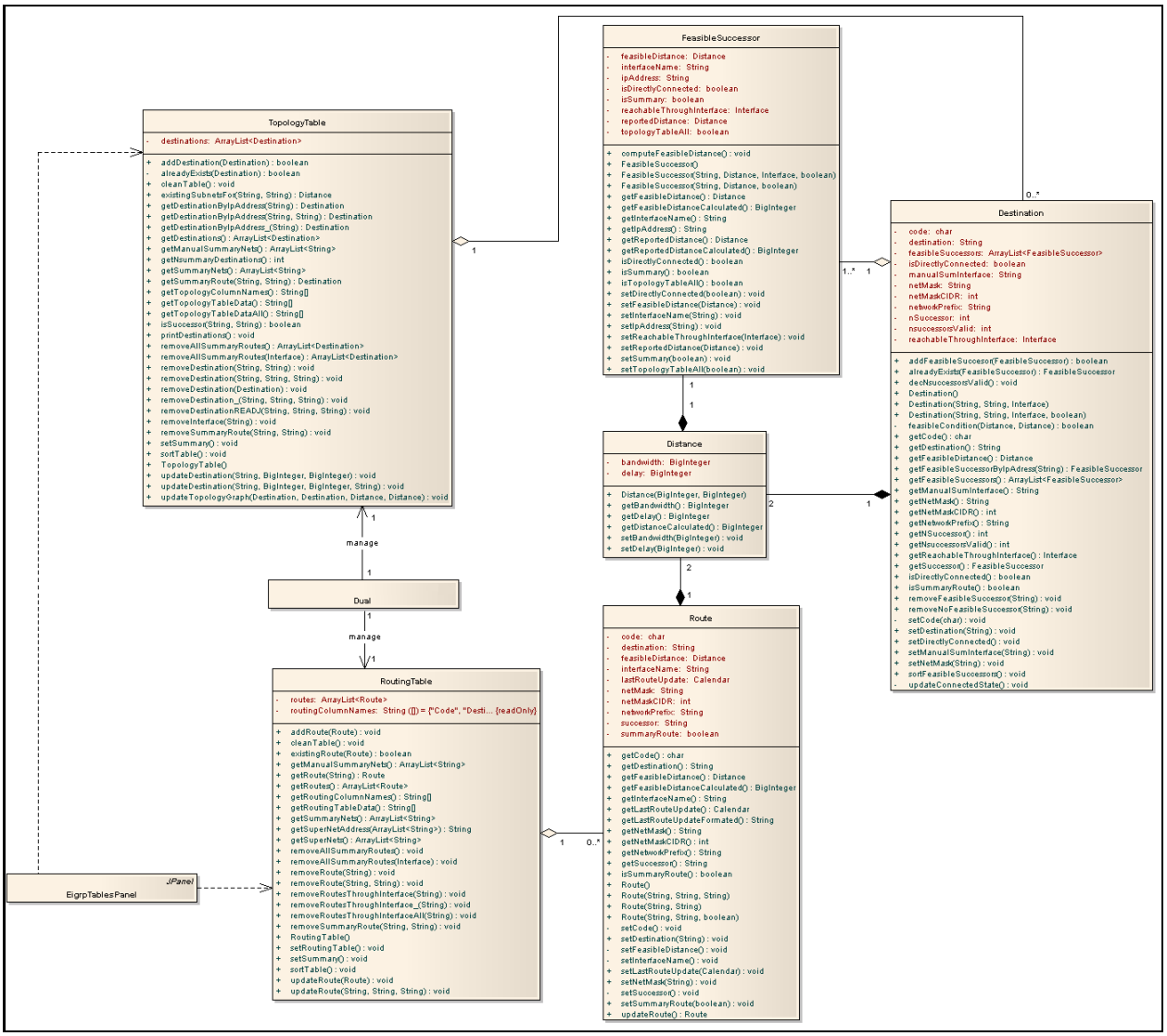

Figura 5.13: Diagrama de Clases General del Módulo *EIGRP Tables* (Parte II)

```
package dataTypes.eigrpTables;

import gui.panels.DualFiniteStateMachinePanel;

public class TopologyTable {

    private static ArrayList<Destination> destinations;

    public TopologyTable(){
        destinations = new ArrayList<Destination>();
    }

    public Destination getDestinationByIpAddress(String ipAddress, String mask){
        for(int i=0; i<destinations.size(); i++){
            Destination dest = destinations.get(i);
            //SUMMARY HERE
            if(dest==null || (Singleton.isAutoSum() && !dest.isSummaryRoute() && dest.isDirectlyConnected()))
                continue;
            //SUMMARY HERE
            if((dest.getDestination().equals(ipAddress)
                    || dest.getNetworkPrefix().equals(Utils.getNetworkPrefix(ipAddress, dest.getNetMask())))
                    && dest.getNetMask().equals(mask)) //SUMMARY HERE
                return dest;
        }
        return null;
    }

    public Destination getDestinationByIpAddress(String ipAddress){

    public Destination getDestinationByIpAddress_(String ipAddress){

    public void removeDestination(String ipAddress, String netMask){

    public void removeDestination(String neighbor, String ipAddress, String netMask){
```

Figura 5.14: Segmento de código para la creación de la Tabla Topológica

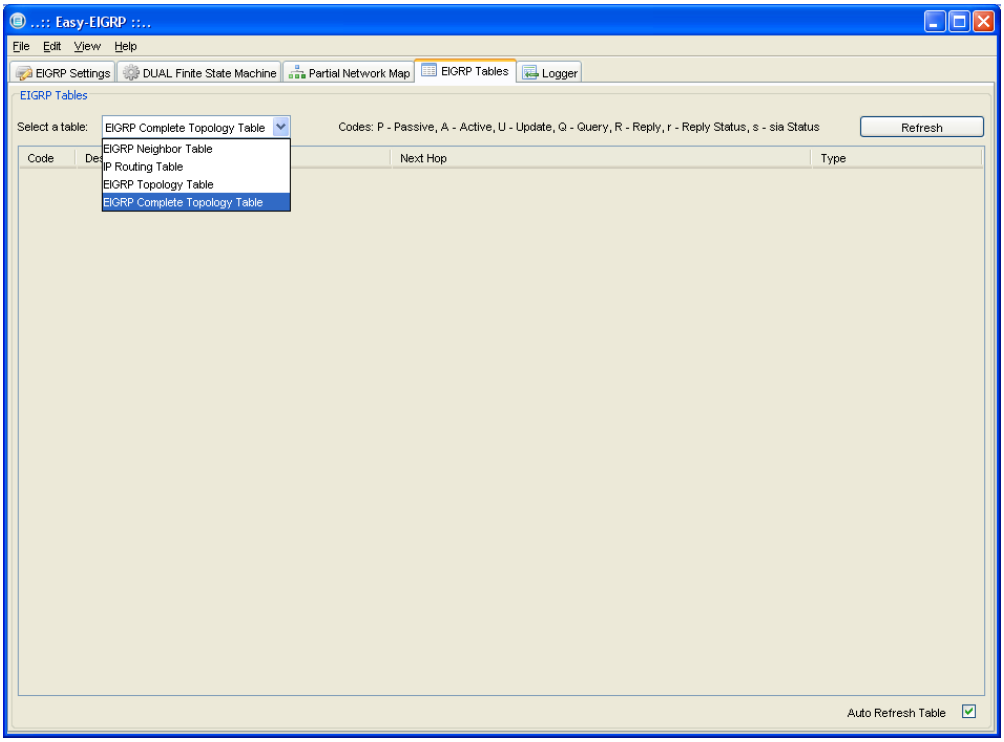

Figura 5.15: Panel *EIGRP Tables*

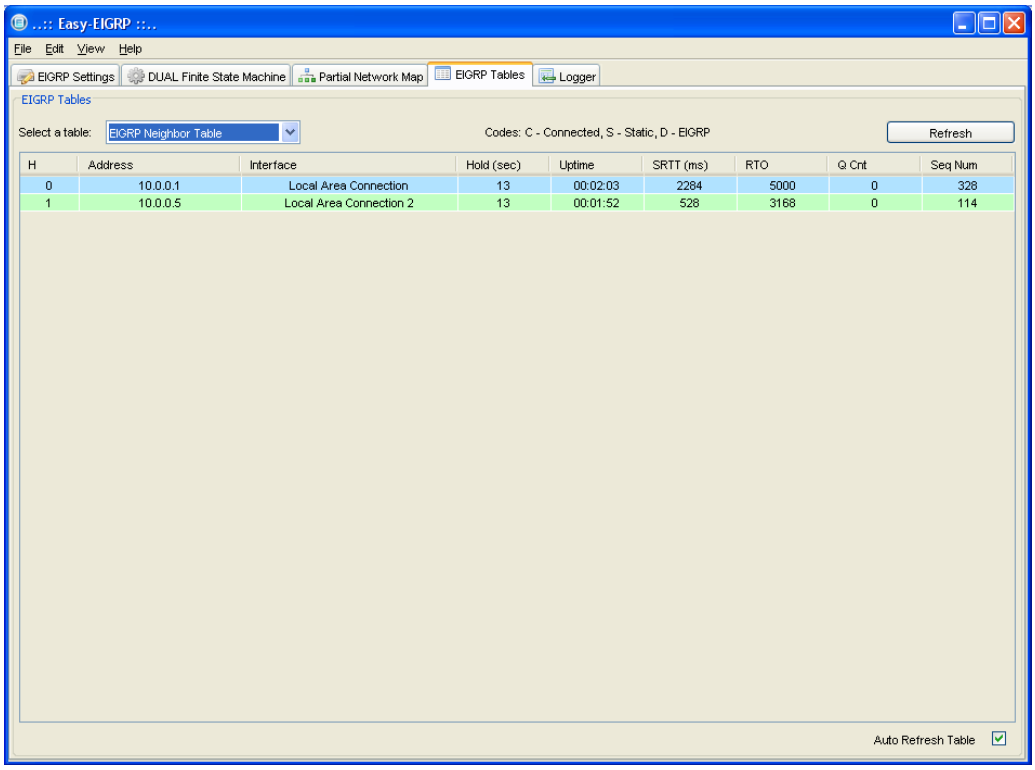

Figura 5.16: Panel *EIGRP Tables*. *EIGRP Neighbor Table*

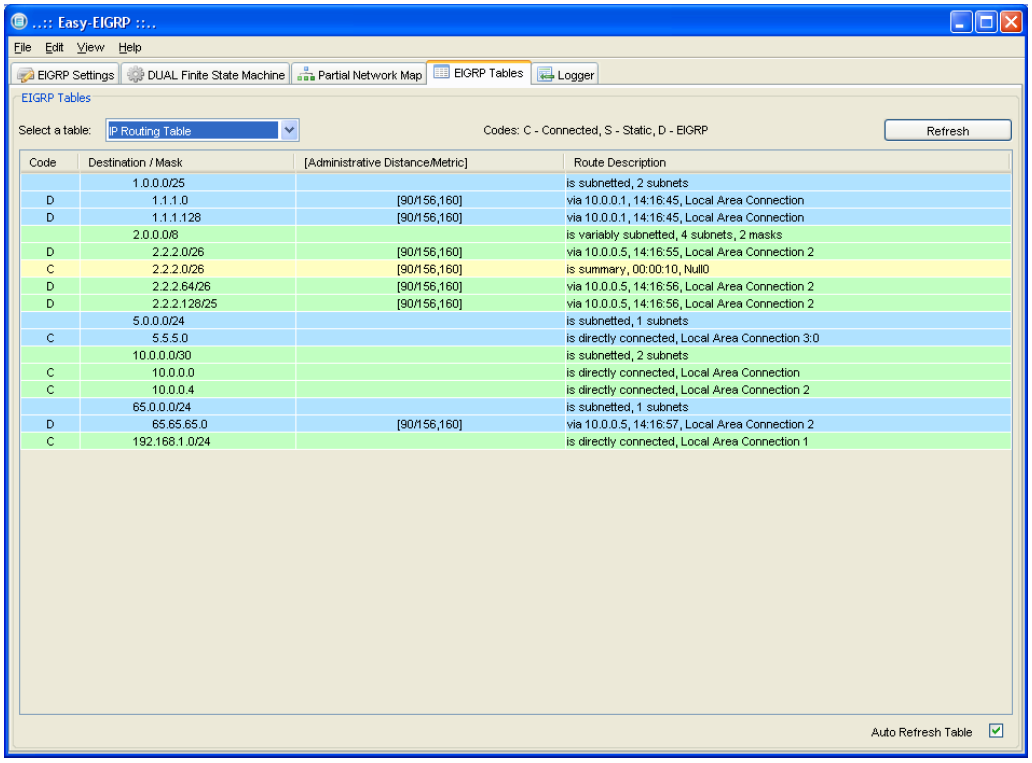

Figura 5.17: Panel *EIGRP Tables*. *IP Routing Table*

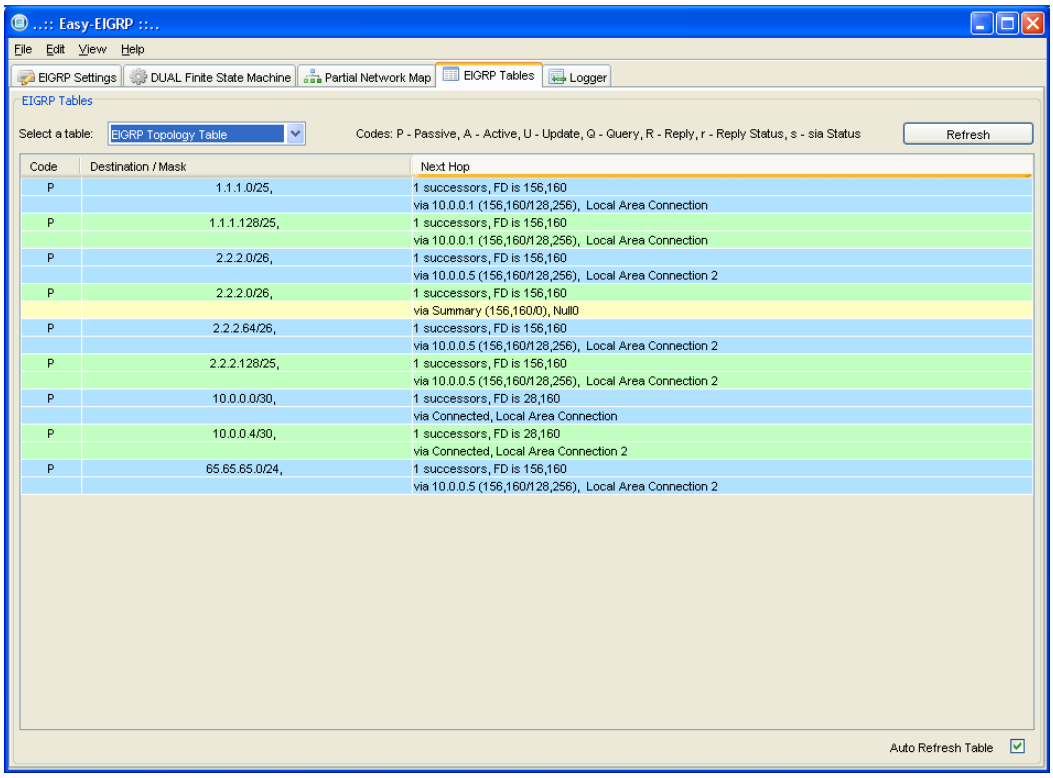

Figura 5.18: Panel *EIGRP Tables*. *EIGRP Topology Table*

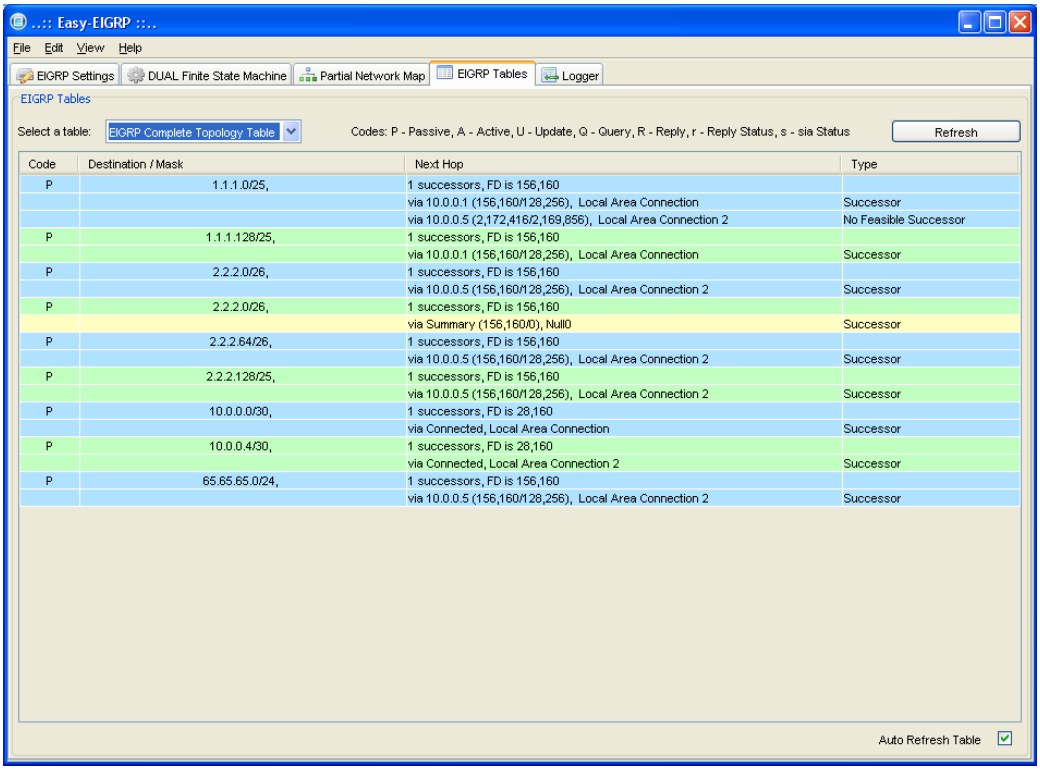

Figura 5.19: Panel *EIGRP Tables*. *EIGRP Complete Topology Table*

5.2.5 *Iteración 5:* Módulo Logger

El siguiente módulo pretende descomponer todos y cada uno de los acontecimientos ocurridos durante la ejecución de la aplicación en mensajes precisos y puntuales para que el usuario tenga conocimiento de todo lo ocurrido. Adicionalmente este panel otorga una herramienta similar a la del *Packet Tracer*[9], con el cual los usuarios pueden observar todos los datos (campos con sus respectivas explicaciones) de los paquetes de EIGRP.

Existen tres niveles de mensajes de depuración: *Debug*, *Info* y *Warn*, los cuales son descritos a continuación.

- *Debug*: son de color verde y muestran el intercambio de los paquetes de EIGRP (*Hello, Acknowledgment, Update, Query* y *Reply*).
- *Info*: son de color azul y proveen información general acerca de los cambio EIGRP (nuevas adyacencias, nuevos prefijos, etc.).
- *Warn*: son de color rojo y están relacionados con la perdida de rutas, cambios inesperados de la interfaz de red, pérdida de paquetes EIGRP, etc.

Adicionalmente, el *logger* cuenta con una serie de filtros que permiten la selección de de los mensajes por tipo de paquete, por alguna entrada personalizada del usuario o por interfaz de red.

5.2.5.1 Fase de análisis

El desarrollo de este módulo debe permitir al usuario observar de manera fácil y eficiente cada uno de los mensajes presentados, para ello la interfaz gráfica debe proveer las herramientas necesarias para cumplir con este objetivo.

Una vez seleccionado un paquete EIGRP se deben mostrar los campos del mismo y los valores de cada uno de los campos en hexadecimal.

5.2.5.2 Fase de Diseño

Para el diseño de este módulo se definieron una serie de relaciones entre cada uno de los módulos para poder obtener una comunicación total y poder almacenar cada uno de las operaciones de la aplicación que luego serán mostradas al usuario. Las principales relaciones pueden observarse en la Figura 5.20.

Por otro lado el diseño de la interfaz gráfica para este módulo se basa en un panel que se encuentra dividido en cuatro secciones principales:

- *Logger*: sección que permite listar cada uno de los mensajes y mostrarlos con su color respectivo en una tabla.
- *Fields Tree:* esta sección contiene un componente gráfico llamado *JTree* que muestra cada uno de los campos del paquete EIGRP.

[9] http://www.cisco.com/web/learning/netacad/course_catalog/PacketTracer.html

- *Bytes Table:* es una tabla que muestra byte a byte cada paquete EIGRP en forma hexadecimal.
- *Logger Filter*: es una sección que proporciona al usuario la posibilidad de filtrar los mensajes según el nivel (*Debug, Info, Warn*) del mensaje, el tipo de paquete EIGRP, la interfaz de red o alguna entrada específica.

Cada una de estas secciones se pueden observar en la Figura 5.21.

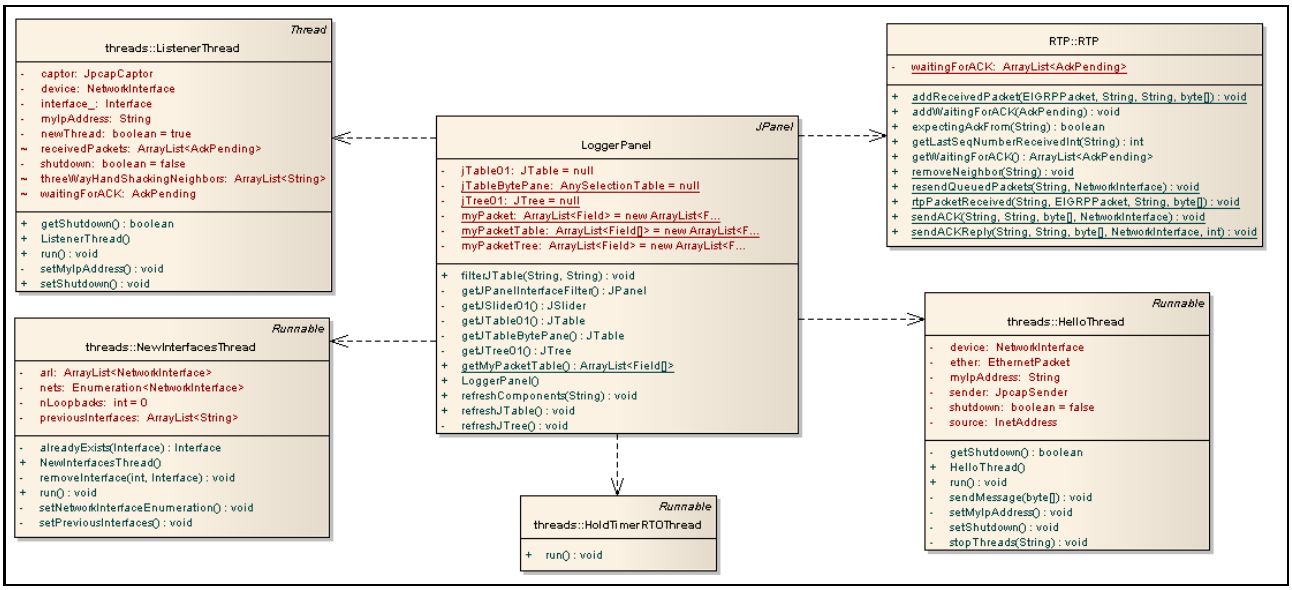

Figura 5.20: Diagrama de Clases General del Módulo *Logger*

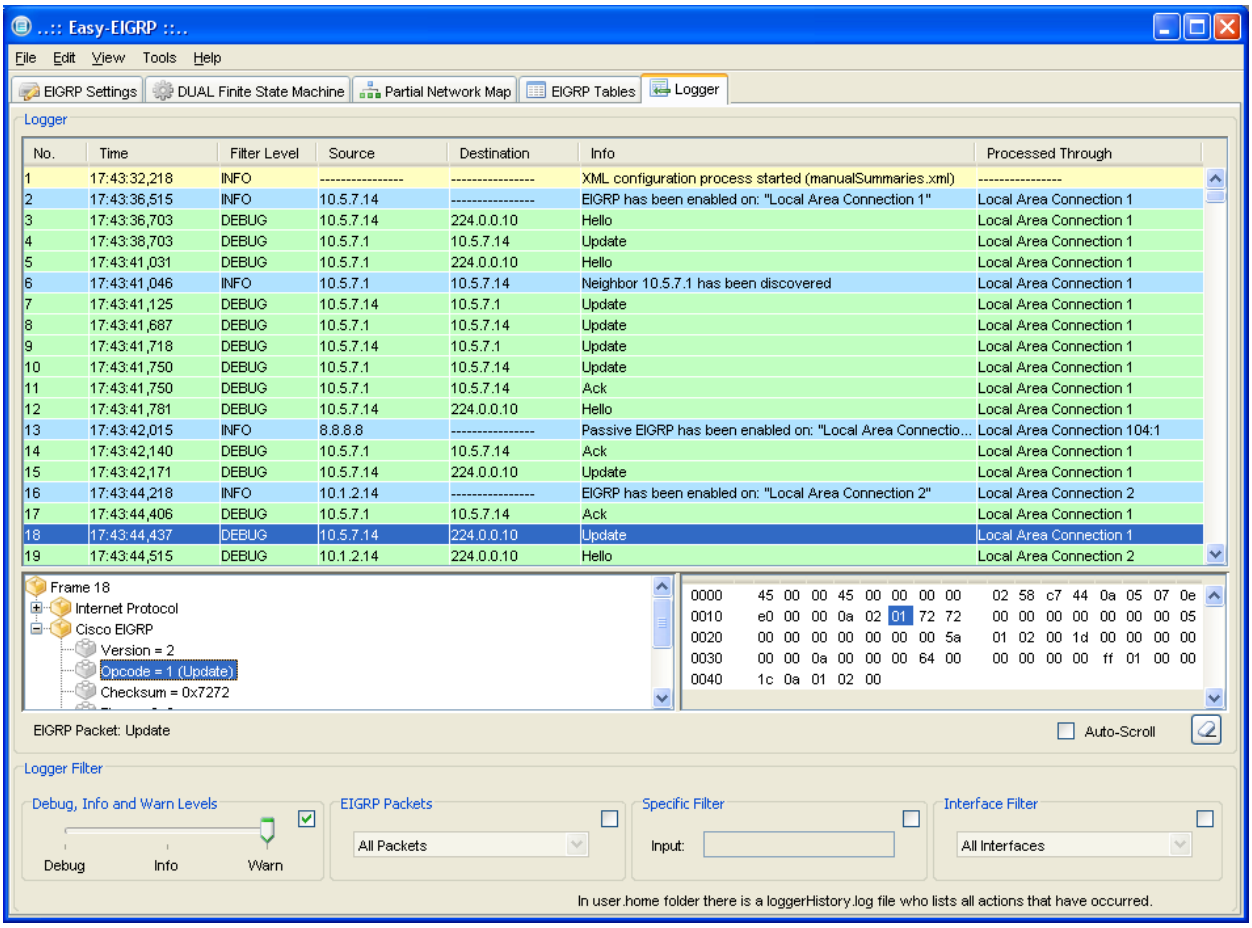

Figura 5.21: Panel *Logger*

5.2.5.3 Fase de Codificación

Para la implementación de este módulo se siguió el patrón de diseño *Singleton*, así como cada una de las pautas presentadas en la fase de diseño.

La interfaz gráfica está compuesta por varios componentes visuales (*JTable*, *JTree*, *JSplitPane*, *JSlider*, etc.) divididos en cuatro paneles principales, en el siguiente segmento de código (Figura 5.22) se puede observar la tabla principal que mostrara cada uno de los mensajes y que el usuario puede seleccionar para ver en detalle su composición en caso de ser un paquete EIGRP.

5.2.5.4 Fase de Pruebas

Debido a que este módulo muestra cada una de las operaciones realizadas, es de gran ayuda para verificar el correcto funcionamiento de la aplicación en un contexto general. Por lo tanto para probarlo simplemente se generaban entradas en la aplicación, edición de interfaces, creación de adyacencias, entre otros eventos, para comprobar que el listado de los mensajes fuese correcto.

```
private JTable getJTable01() {
    if (jTable01 == null) {
        jTable01 = new JTable();
        jTable01.setName("LoogerPanel");
        //jTable01.setAutoResizeMode(JTable.AUTO_RESIZE_OFF);
        jTable01.setDefaultRenderer(Object.class, new ColoredTableCellRenderer());
        jTable01.setDefaultRenderer(Integer.class, new ColoredTableCellRenderer());

        jTable01.setModel(new DefaultTableModel(
                new Object [][] {},
                new String [] {
                        "No.","Time", "Filter Level", "Source","Destination","Info","Processed Through"
                }
        ){
            private static final long serialVersionUID = 1L;

            Class[] types = new Class [] {
            boolean[] canEdit = new boolean [] {
            public Class getColumnClass(int columnIndex) {
            public boolean isCellEditable(int rowIndex, int columnIndex) {
        });

        jTable01.setCellSelectionEnabled(false);
        jTable01.setColumnSelectionAllowed(false);
        jTable01.setRowSelectionAllowed(true);
        jTable01.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
        jTable01.setGridColor(Color.white);

        jTable01.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent e) {
                String filterLevel = (String)jTable01.getModel().getValueAt(
                        jTable01.convertRowIndexToModel(jTable01.getSelectedRow())
                        ,2
                );

                String value = (String)jTable01.getModel().getValueAt(
                        jTable01.convertRowIndexToModel(jTable01.getSelectedRow())
```

Figura 5.22: Segmento de código para la creación del *JTable* informativo

Más adelante se pueden observar con más detalle los distintos escenarios de pruebas utilizados para comprobar el correcto comportamiento de cada uno de los módulos que conforman la aplicación.

5.3 Fase de Pruebas y Depuración Final

Una vez finalizadas las iteraciones que hicieron posible el desarrollo de Easy-EIGRP, se procedió a crear múltiples escenarios específicos para así poder realizar pruebas en conjunto y observar el comportamiento de la aplicación y constatar que la interacción entre todos los módulos de la misma sea correcta.

A continuación se muestran los escenarios más relevantes junto con una breve descripción de las pruebas realizadas.

5.3.1 Escenario 1

El primer escenario se basa en una topología muy simple y el objetivo de la misma es experimentar con situaciones básicas y verificar el correcto funcionamiento de la aplicación para luego proceder a evaluar situaciones más complejas.

En la Figura 5.23 puede apreciarse dicho escenario. En el mismo se puede observar que sólo se contará con dos vecinos, cada uno de ellos contará con una interfaz *loopback*. Las métricas de cada enlace se encuentran descritas de color azul donde, el primer valor corresponde al ancho de banda y el segundo al retardo (*delay*).

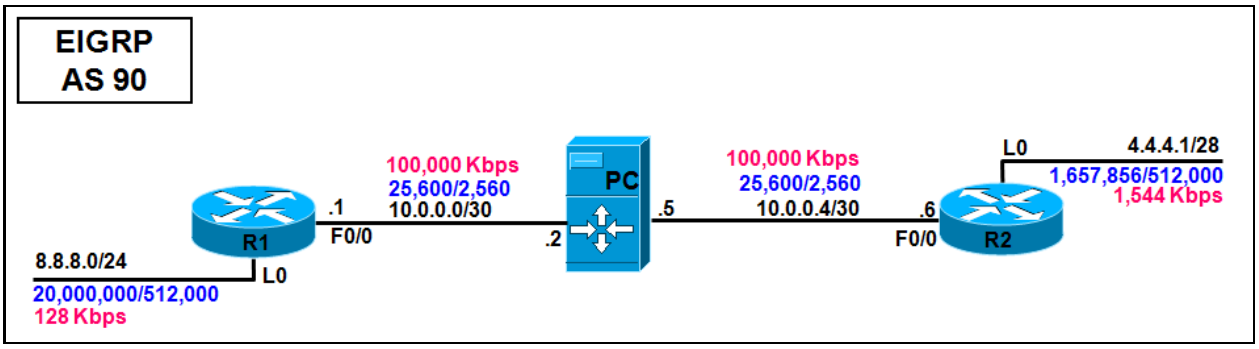

Figura 5.23: Escenario de Pruebas 1

Las pruebas realizadas en esta ocasión fueron las expuestas en la Tabla 5.1.

Descripción de la Prueba	Módulos Afectados	Módulos que Presentaron Fallas
Configuración de las dos interfaces a utilizar del PC, incluyendo cambio de direcciones IP y máscaras	EIGRP Settings EIGRP Tables Logger	-
Inicialización de procesos EIGRP en ambas interfaces	EIGRP Settings EIGRP Tables Partial Network Map Logger	Logger: se tuvo que agregar la cabecera IP para la visualización de los paquetes.
Procesamiento de cambio de métrica anunciada por un vecino	EIGRP Tables Partial Network Map Logger	-
Procesamiento de cambio de métrica de un enlace a través de Easy-EIGRP	EIGRP Settings EIGRP Tables Partial Network Map Logger	-
Procesamiento de pérdida de un prefijo de red por parte de un vecino	EIGRP Tables Partial Network Map Logger	-
Procesamiento de conocimiento de un nuevo prefijo de red por parte de un vecino	EIGRP Tables Partial Network Map Logger	-
Pérdida de un vecino	EIGRP Tables Partial Network Map Logger	-
Conocimiento de un nuevo vecino	EIGRP Tables Partial Network Map Logger	-

Pérdida de una interfaz EIGRP en la aplicación	EIGRP Settings EIGRP Tables Partial Network Map Logger	-

Tabla 5.1: Tabla de Depuración 1

5.3.2 Escenario 2

El segundo escenario (ver Figura 5.24) es muy similar al presentado en la Sección 5.3.1, sin embargo en este caso se incluyeron dos interfaces *loopback* a Easy-EIGRP.

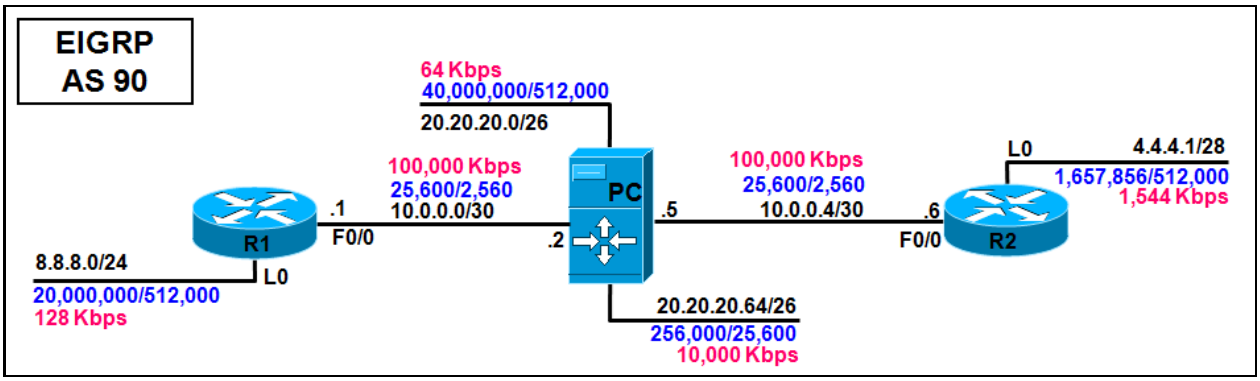

Figura 5.24: Escenario de Pruebas 2

En la Tabla 5.2 se presentan las pruebas realizadas.

Descripción de la Prueba	Módulos Afectados	Módulos que Presentaron Fallas
Configuración de las dos interfaces a utilizar del PC, incluyendo cambio de direcciones IP y máscaras	EIGRP Settings EIGRP Tables Logger	-
Creación de dos interfaces *loopback*	EIGRP Settings EIGRP Tables	EIGRP Settings: existieron inconvenientes con la asignación de direcciones IP
Inicialización de procesos EIGRP tanto en las interfaces físicas como las *loopback*	EIGRP Settings EIGRP Tables Partial Network Map Logger	-
Cambio de métrica de una interfaz *loopback* de Easy-EIGRP	EIGRP Tables Partial Network Map Logger	-
Pérdida de una interfaz *loopback* de Easy-EIGRP	EIGRP Settings EIGRP Tables Partial Network Map Logger	-

Procesamiento de cambio de métrica anunciada por un vecino	EIGRP Tables Partial Network Map Logger	-
Cambio de métrica de un enlace a través de Easy-EIGRP	EIGRP Settings EIGRP Tables Partial Network Map Logger	-
Pérdida de un prefijo de red por parte de un vecino	EIGRP Tables Partial Network Map Logger	-
Conocimiento de un nuevo prefijo de red por parte de un vecino	EIGRP Tables Partial Network Map Logger	-
Pérdida de un vecino	EIGRP Tables Partial Network Map Logger	-
Conocimiento de un nuevo vecino	EIGRP Tables Partial Network Map Logger	-
Pérdida de una interfaz EIGRP en la aplicación	EIGRP Settings EIGRP Tables Partial Network Map Logger	-
Summarization automática en un vecino	EIGRP Settings EIGRP Tables Partial Network Map Logger	-
No *summarization* en un vecino con previa *summarization* automática	EIGRP Settings EIGRP Tables Partial Network Map Logger	-
Summarization automática en Easy-EIGRP	EIGRP Settings EIGRP Tables Partial Network Map Logger	-
Eliminación de *summarization* automática en Easy-EIGRP	EIGRP Settings EIGRP Tables Partial Network Map Logger	-
Summarization Manual en Easy-EIGRP	EIGRP Settings EIGRP Tables Partial Network Map Logger	EIGRP Tables: existió un problema en el intercambio de paquetes lo cual conllevaba a un estado incorrect de las tablas EIGRP

Eliminación de *Summarization* Manual en Easy-EIGRP	EIGRP Settings EIGRP Tables Partial Network Map Logger	EIGRP Tables: existió un problema en el intercambio de paquetes lo cual conllevaba a un estado incorrecto de las tablas EIGRP
Cambio de los valores K's tanto en Easy-EIGRP como en los vecinos	EIGRP Settings EIGRP Tables Partial Network Map Logger	EIGRP Tables: existió un inconveniente a la hora de mostrar las métricas de los prefijos en las tablas

Tabla 5.2: Tabla de Depuración 2

5.3.3 Escenario 3

La finalidad del escenario diseñado, el cual puede apreciarse en la Figura 5.25, es la de observar el comportamiento de la aplicación al momento de manejar múltiples vecinos y procesar múltiples intercambios de paquetes por una misma interfaz de red.

En este caso se cuenta con 3 vecinos, cada uno de ellos posee una interfaz *loopback* configurada. Nuevamente, en color azul se encuentran los pesos respectivos para cada enlace.

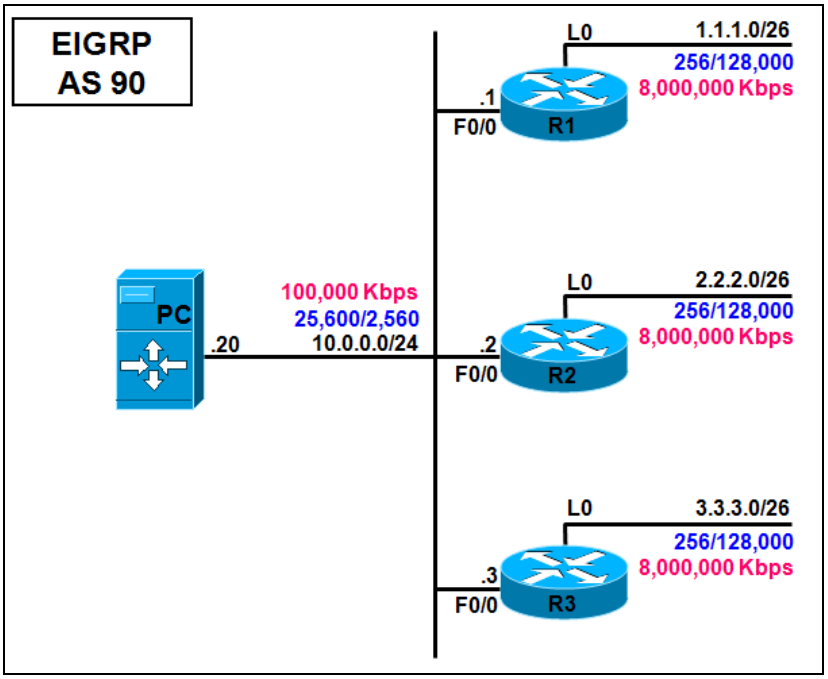

Figura 5.25: Escenario de Pruebas 3

Las pruebas y resultados obtenidos para el escenario 3 son las que se muestran en la Tabla 5.3.

Descripción de la Prueba	Módulos Afectados	Módulos que Presentaron Fallas
Configuración de la interfaz a utilizar del PC, incluyendo cambio de dirección IP y máscara	EIGRP Settings EIGRP Tables Logger	-
Inicialización del proceso EIGRP en la interfaz física	EIGRP Settings EIGRP Tables Partial Network Map Logger	-
Procesamiento de cambio de métrica anunciada por un vecino	EIGRP Tables Partial Network Map Logger	-
Cambio de métrica del enlace a través de Easy-EIGRP	EIGRP Settings EIGRP Tables Partial Network Map Logger	-
Pérdida de un prefijo de red por parte de un vecino	EIGRP Tables Partial Network Map Logger	-
Conocimiento de un nuevo prefijo de red por parte de un vecino	EIGRP Tables Partial Network Map Logger	-
Pérdida de un vecino	EIGRP Tables Partial Network Map Logger	-
Conocimiento de un nuevo vecino	EIGRP Tables Partial Network Map Logger	-
Pérdida de una interfaz EIGRP en la aplicación	EIGRP Settings EIGRP Tables Partial Network Map Logger	-
Summarization automática en un vecino	EIGRP Settings EIGRP Tables Partial Network Map Logger	-
Eliminación de *summarization* automática en un vecino	EIGRP Settings EIGRP Tables Partial Network Map Logger	-
Summarization automática en Easy-EIGRP	EIGRP Settings EIGRP Tables Partial Network Map Logger	-

Eliminación de *summarization* automática en Easy-EIGRP	EIGRP Settings EIGRP Tables Partial Network Map Logger	-
Cambio de los valores K's tanto en Easy-EIGRP como en los vecinos	EIGRP Settings EIGRP Tables Partial Network Map Logger	Partial Network Map: las imágenes de los vecinos se superponían en vez de recordar sus posiciones anteriores

Tabla 5.3: Tabla de Depuración 3

5.3.4 Escenario 4

El escenario que se muestra en la Figura 5.26 fue diseñado con el objetivo de hacer pruebas bajo condiciones más complejas y más reales. Además, se cuenta con dos formas de llegar a la *subnet* 5.5.5.0/25 (por el R2 y por R5), de esta forma puede llevarse a cabo pruebas de pérdida de *successors* y observar el comportamiento del módulo *DUAL Finite State Machine*.

Para la Figura 5.26 se realizaron las pruebas que se muestran en la Tabla 5.4.

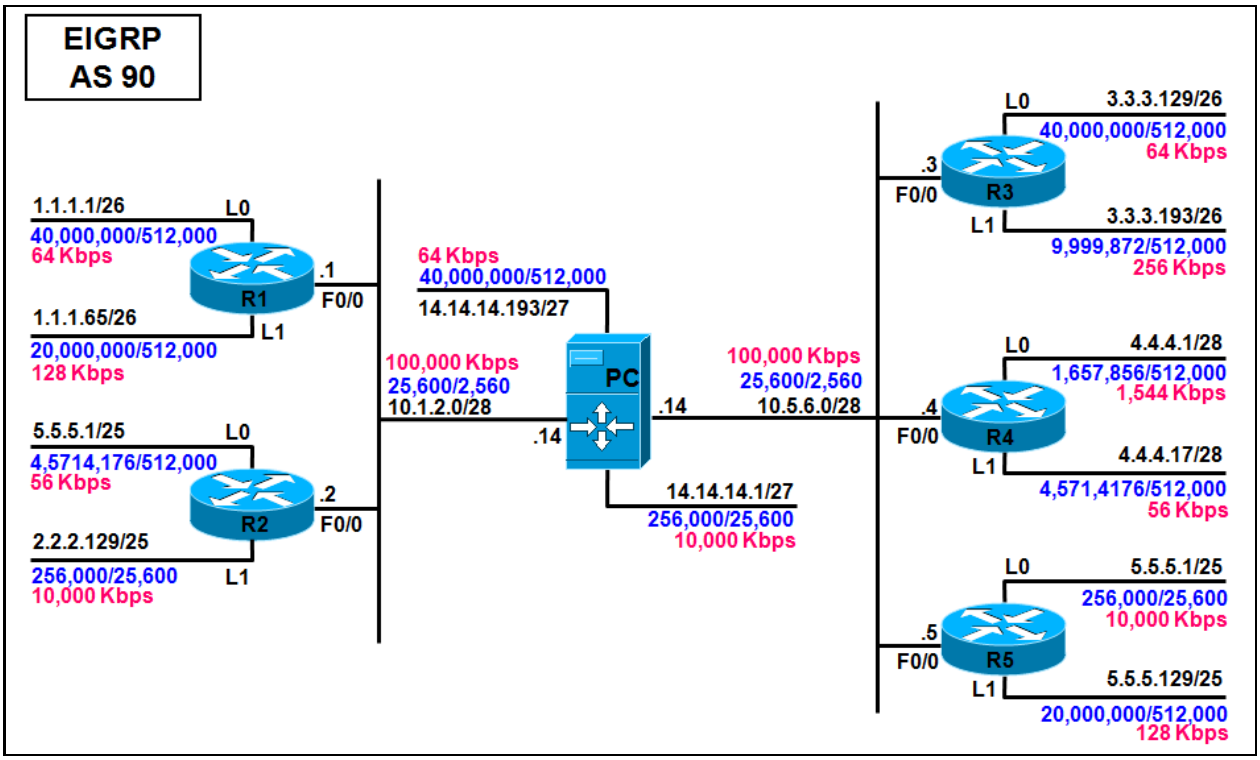

Figura 5.26: Escenario de Pruebas 4

Descripción de la Prueba	Módulos Afectados	Módulos que Presentaron Fallas
Configuración de la interfaces a utilizar del PC e inicialización del proceso EIGRP en la interfaz física	EIGRP Settings EIGRP Tables Partial Network Map Logger	-
Procesamiento de cambio de métrica anunciada por un vecino	EIGRP Tables Partial Network Map Logger	-
Cambio de métrica del enlace a través de Easy-EIGRP	EIGRP Settings EIGRP Tables Partial Network Map Logger	-
Pérdida y conocimiento de un prefijo de red por parte de un vecino	EIGRP Tables Partial Network Map Logger	-
Pérdida y conocimiento de un vecino	EIGRP Tables Partial Network Map Logger	-
Pérdida de una interfaz EIGRP en la aplicación	EIGRP Settings EIGRP Tables Partial Network Map Logger	-
Summarization automática y eliminación de *summarization* automática en un vecino	EIGRP Settings EIGRP Tables Partial Network Map Logger	-
Summarization automática y eliminación de *summarization* automática en Easy-EIGRP	EIGRP Settings EIGRP Tables Partial Network Map Logger	-
Cambio de los valores K's tanto en Easy-EIGRP como en los vecinos	EIGRP Settings EIGRP Tables Partial Network Map Logger	-
Pérdida de un prefijo teniendo *feasible successors*	EIGRP Tables Partial Network Map Logger DUAL Finite State Machine	DUAL Finite State Machine: el temporizador de tiempo para determinar rutas SIA no funcionaba correctamente

Pérdida de un prefijo sin *feasible successors*	EIGRP Tables Partial Network Map Logger DUAL Finite State Machine	DUAL Finite State Machine: el temporizador de tiempo para determinar rutas SIA no funcionaba correctamente. Se agregó información en la interfaz de usuario para explicar el evento ocurrido
Cambio del valor de la Varianza	EIGRP Settings EIGRP Tables	-

Tabla 5.4: Tabla de Depuración 4

5.3.5 Escenario 5

El escenario 5, detallado en la Figura 5.27, es muy similar al escenario de la Sección 5.3.4, la diferencia radica en la utilización de una interfaz física del computador adicional al igual que dos interfaces *loopback*.

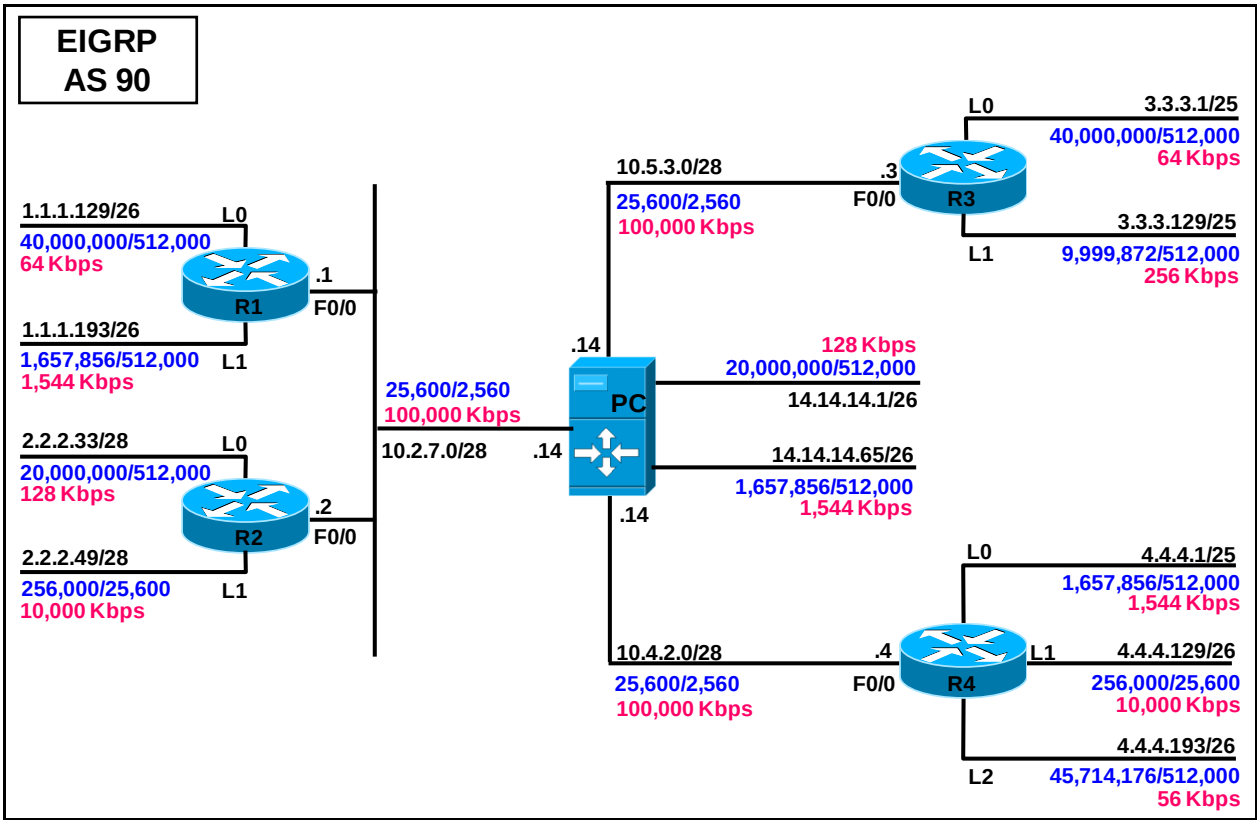

Figura 5.27: Escenario de Pruebas 5

Las pruebas realizadas en este caso fueron las mismas que se realizaron en los escenarios anteriores. Para ninguna de las pruebas se encontró ningún módulo que presentara algún tipo de falla.

6. Especificaciones Técnicas de Easy-EIGRP

Easy-EIGRP fue desarrollada en su mayoría en el Laboratorio de Internet II, ubicado en la Facultad de Ciencias de la Universidad Central de Venezuela.

Dicho laboratorio posee un total de 6 routers manufacturados por Cisco Systems, además de múltiples hubs y switchs para la creación de diferentes topologías según fuera necesario. En el laboratorio se encuentran alrededor de 10 computadores marca *Sun Microsystems*[10], las cuales cuentan con 2 procesadores AMD Opteron 246 (2 GHz) y 2 GB de memoria RAM. Los computadores utilizados cuentan con al menos 3 tarjetas de red 10/100/1000-Mbps base-T (Gigabit Ethernet).

Easy-EIGRP fue desarrollado en su mayoría utilizando *Windows XP Professional*[11] en inglés y utilizando *Eclipse IDE for Java Developers*[12] como entorno de desarrollo. Sin embargo, también fue probado y en algunas ocasiones modificado utilizando las versiones más actualizadas de *Ubuntu*[13] en inglés.

Todas las pruebas de desarrollo se hicieron utilizando routers de Cisco Systems y utilizando topologías simuladas con GNS3 (*Graphical Network Simulator 3*)[14]. Dado que la versión de IOS utilizada para desarrollar Easy-EIGRP fue la 12.4 y la versión de EIGRP implementada fue la 1.2, para GNS3 se utilizó la imagen de IOS *c3640-ik9o3s-mz.124-7*.

Con respecto a las dependencias de Easy-EIGRP, esta fue desarrollada y probada con Jpcap 0.7 y WinPcap 4.1.1.

Easy-EIGRP debe ser ejecutado en modo administrador (*Windows*) o *root* (*Linux*).

Para lograr que el computador se comporte como router, es necesario que el usuario ejecute los siguientes pasos antes de iniciar Easy-EIGRP:

- Windows:
  - Iniciar el Editor de Registros de Windows (Regedit.exe).
  - Localizar el siguiente Regitry Key: *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters*
  - Configurar los siguientes *keys* con los valores que se muestran a continuación:

Value Name: IPEnableRouter
Value type: REG_DWORD
Value Data: 1

[10] http://www.oracle.com
[11] http://www.microsoft.com/windowsxp/pro/default.mspx
[12] http://www.eclipse.org
[13] http://www.ubuntu.com
[14] http://www.gns3.net

- Abandonar Editor de Registros de Windows.

- Linux:
  - Ejecutar el siguiente comando en la consola:

echo 1 > /proc/sys/net/ipv4/ip_forward

En la Tabla 6.1 se presenta un resumen de las especificaciones antes expuestas.

Característica Técnica	**Especificación**
Sistemas Operativos	Windows (XP Professional, recomendado) Linux (Ubuntu, recomendado)
Procesador	La pruebas fueron realizadas en: - AMD Opteron 246 - Intel Celeron 550 - AMD Athlon 3000/3200+
Cantidad de RAM	1 GB mínimo (Recomendado)
Cisco IOS	12.4
Versión EIGRP	1.2
Imagen IOS para GNS3	c3640-ik9o3s-mz.124-7
Jpcap	Jpcap 0.7 (Recomendado)
WinPcap	WinPcap 4.1.1 (Recomendado)

Tabla 6.1: Especificaciones Técnicas de Easy-EIGRP

7. Conclusiones

EIGRP nace con el fin de cubrir las necesidades básicas de enrutamiento de redes, haciéndolo de la forma más eficiente y eficaz posible, incluyendo la menor utilización de recursos, la mayor rapidez y el mantenimiento de estados permanentes de coherencia.

Analizando el estudio realizado sobre EIGRP, puede observarse que existen numerosas y significantes ventajas que dicho protocolo trae consigo. Sin embargo y debido a que EIGRP es patentado, realizar actividades como estudios detallados, modificaciones de los mecanismos del protocolo es básicamente imposible o de gran dificultad.

Como consecuencia de las razones anteriormente expuestas y haciendo uso de la ingeniería inversa, se ha desarrollado una herramienta multiplataforma, denominada Easy-EIGRP, la cual implementa EIGRP (protocolo de enrutamiento de vector de distancia avanzado) sobre redes IP.

Para su implementación fue utilizado Java como lenguaje de programación, el cual permitió cubrir los objetivos planteados como Trabajo Especial de Grado.

El lenguaje de programación utilizado fue complementado con algunas librerías de código abierto, las cuales tienen como finalidad ampliar las capacidades de Java. Dichas librerías facilitaron el desarrollo de los módulos provistos por la aplicación. Entre las librerías utilizadas se encuentran:

- Jpcap.
- WinPcap.
- Libpcap.
- Devcon.
- JUNG.
- Log4j.

Easy-EIGRP es una limitada pero poderosa herramienta para el aprendizaje y la enseñanza del protocolo EIGRP la cual también puede ser utilizada en una red de producción. La aplicación posee una cantidad significativa de ventajas, entre las cuales se destacan:

- Entorno amigable, intuitivo e interactivo el cual facilita el proceso de entendimiento de EIGRP.
- Facilidad para el instructor a la hora de explicar cualquier aspecto o proceso del protocolo gracias a la sencillez y el nivel de abstracción utilizado al momento de desarrollar la aplicación, sobre todo a nivel de interfaz gráfica.
- Amplia gama de herramientas gráficas para entender cualquier aspecto del protocolo, incluyendo gráficos, tablas, etc.
- A diferencia del CLI (*Command Line Interface*) de Cisco Systems, Easy-EIGRP provee la información del protocolo de forma clara y natural.

- Estudio del estado del router basado en PC en un instante de tiempo dado (depuración paso a paso) gracias a las posibilidades de configuración de auto-refrescamiento de tablas, de repetición de eventos, de almacenamiento de paquetes y/o mensajes intercambiados, entre otros.
- Almacenamiento de configuraciones de Easy-EIGRP a través de archivos XML[15], (*Extensible Markup Language*) ahorrando así tiempo en el estudio de algún escenario recurrente.
- Independencia de equipos de Cisco Systems u otras herramientas para la simulación de routers EIGRP.
- Portabilidad entre ambientes Windows y Linux.

Easy-EIGRP constituye una herramienta ideal para ser utilizada a nivel didáctico en universidades o cursos avanzados, como por ejemplo el CCNA (*Cisco Certified Network Associate*) o el CCNP (*Cisco Certified Network Professional*), donde con seguridad se logrará mayor interés y entendimiento por parte de los estudiantes gracias a las ventajas que Easy-EIGRP trae consigo.

Finalmente, es necesario mencionar que a pesar de que todos los objetivos propuestos al inicio de la investigación fueron alcanzados con éxito, se considera que existen ciertos trabajos complementarios que enriquecerían significativamente la aplicación. Dichos trabajos son los recomendados a continuación:

- Manejo de más de un Sistema Autónomo.
- Implementación completa de la máquina de estados finitos DUAL.
- Proveer autenticación MD5.

[15] http://www.w3.org/XML

Referencias Bibliográficas

[1] C. Hedrick. Routing Information Protocol. RFC 1058. Junio, 1988.

[2] CCNA Exploration 4.0. Cisco Systems.

[3] D. Comer. Redes Globales de Información con Internet y TCP/IP, Principios básicos, protocolos y arquitectura, Tercera Edición. Prentice Hall. 1996.

[4] D. Fishburne. EIGRP Deployment. Cisco Networkers. 2006.

[5] D. Heywood. Novell's Guide to NetWare® 5 and TCP/IP. Hungry Minds. Marzo, 1999.

[6] Digital Equipment Corporation. DECnet/OSI: The Foundation for Open Networking. DEC. 1990.

[7] E. Rosen. Exterior Gateway Protocol. RFC 827. Octubre, 1982.

[8] EIGRP White Papers. Cisco Systems.

[9] G. Malkin, R. Minnear. RIPng for IPv6. RFC 2080. Enero, 1997.

[10] G. Malkin. A MIME Content-Type for Directory Information. RFC 2453. Noviembre, 1998.

[11] G. Malkin. RIP Version 2 Carrying Additional Information. RFC 1723. Noviembre, 1994.

[12] G. Malkin. RIP Version 2. RFC 2453. Noviembre, 1998.

[13] G. Sackett. Manual de routers Cisco. McGraw-Hill. 2002.

[14] I. Pepelnjka. EIGRP Network Design Solutions. Cisco Press. 2000.

[15] IEEE. 802.2 Logical Link Control. IEEE. 1998.

[16] J. Doyle. Routing TCP/IP. Volume I. Agosto, 2001.

[17] J. Forster, G. Satz, G. Glick, R. Day. Cisco Systems X.25 over TCP. RFC 1613. Mayo, 1994.

[18] J. Halpern, S. Bradner. RIPv1 Applicability for Historic Status. RFC 1923. Marzo, 1996.

[19] J. Moy. OSPF Version 2. RFC 2328. Abril, 1998.

[20] L. Ochaeta. BSCI Módulo 2 – Lección 1 de 1. EIGRP. 2004.

[21] P. Gross. Choosing a “Common IGP” for the IP Internet. RFC 1371. Octubre, 1992.

[22] R. Rivest. The MD5 Message-Digest Algorithm. RFC 1321. Abril, 1992.

[23] S. Randesi. Sna: IBM's Systems Network Architecture. VNR Computer Library. Julio, 1992.

[24] S. Waldbusser. AppleTalk Management Information Base. RFC 1243. Julio, 1991.

[25] T. Li, R. Atkinson. System to Intermediate System (IS-IS). RFC 3567. Julio, 2003.

[26] T. Shaughnessy, T. Velte. Manual Cisco. McGraw-Hill. 2000.

[27] Y. Rekhter, T. Li. A Border Gateway Protocol 4 (BGP-4). RFC 1771. Marzo, 1995.

[28] JUNG: Java Universal Network Graph Framework. http://jung.sourceforge.net.