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

Diseño e Implementación de un Protocolo de Enrutamiento de Vector de Distancia Basado en el Retardo

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

Daniel H. Gámez V.
V-15.488.189
daniel.gamez@gmail.com

Paúl D. Marrero F.
V-14.323.466
paul.marrero@gmail.com

para optar al título de Licenciado en Computación

Tutor**:** Prof. Eric Gamess

Caracas, Noviembre 2011

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 Daniel Humberto Gámez. C.I.: 15.488.189 y Paúl David Marrero C.I.: 14.323.466, con el título "**Diseño e Implementación de un Protocolo de Enrutamiento de Vector de Distancia Basado en el Retardo**", 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 04 de Noviembre de 2011, a las 10:00 AM, para que sus autores lo defiendan en forma pública, en Sala I 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 cuatro días del mes de Noviembre del año dos mil once, dejándose también constancia de que actuó como Coordinador del Jurado el Profesor Tutor Eric Gamess.

Prof. Eric Gamess
(Tutor)

Prof. Héctor Navarro
(Jurado Principal)

Profa. María Elena Villapol
(Jurado Principal)

AGRADECIMIENTOS

Este Trabajo Especial de Grado y todo el esfuerzo que amerita culminar esta espléndida carrera se lo dedico especialmente a mi madre Teresa, quien ha puesto de todo su ser para que yo obtenga este logro.

También comparto este agradable mérito con las personas que siempre han confiado en mí y me han inspirado para alcanzar esta meta, mi hermana Reyther quien ha sido mi norte, mi bella sobrina Alejandra a quien adoro, mi amada esposa Rosangela a quien agradezco su inmensa paciencia y apoyo, a mi familia venezolana, a mi familia madeirense y a todos mis amigos quienes hacen de la vida una gran celebración.

Sin duda agradezco enormemente a mi querida U.C.V., a sus excepcionales profesores, a su invaluable y hermosa infraestructura patrimonio cultural de la humanidad y a quienes la hicieron posible, a la comunidad que la conforma, siento un enorme orgullo de llevarla en mi corazón, seré Ucevista de por vida.

A mi país Venezuela, quien me ha brindado la oportunidad de demostrarle al mundo de nuestro inmenso potencial.

Daniel Gámez.

AGRADECIMIENTOS

A mi madre Deyanira, por su incansable afán y que con empeño nos da todo con enseñanzas, valores, infinito amor y disciplina.

A mi abuelo Ovidio, por su cariño y estar siempre pendiente de lo que todos necesitemos.

A mi abuela Griselia, por sus lecciones de vida, consejos, cariño y su grandioso sentido del humor, te extraño.

A mi hermano Sergio, por enseñarme que con esfuerzo se puede lograr lo imposible, te quiero Chamaco.

A mi tío César, por el cariño, apoyo y el excelente humor que lo caracteriza.

A mi prima Nailyn, por su apoyo.

A Karen, por apoyarme y comprender el sacrificio que hacemos para forjar un porvenir lleno de logros.

A mi tutor Eric Gamess, por su exigencia, desmedida dedicación y calidad docente.

A todos aquellos familiares y amigos que colaboraron en este proyecto.

Paul D. Marrero F.

RESUMEN

Título:
Diseño e Implementación de un Protocolo de Enrutamiento de Vector de Distancia Basado en el Retardo

Autores:
Daniel Gámez, Paúl Marrero.

Tutor:
Prof. Eric Gamess.

En este Trabajo Especial de Grado se propone un nuevo protocolo de enrutamiento llamado DBRP (*Delay Based Routing Protocol*). DBRP es un flexible protocolo de vector de distancia con métrica basada en el retardo. Nativamente soporta IPv4 e IPv6 y es capaz de transportar información de servicios comunes tales como direcciones IP de servidores DNS. Esta última característica lo hace único, dado que los protocolos de enrutamiento existentes se limitan a transportar información de enrutamiento. DBRP también soporta autenticación y cifrado. Éste opera sobre la capa de enlace y está basado en una arquitectura de tuplas TLV (*Type-Length-Value*); por lo cual puede ser fácilmente extendido a otros protocolos de red existentes o futuros. Adicionalmente, al definir nuevas tuplas TLV, otros servicios comunes como NTP, NIS, NIS+, SIP, Log, y servidores de impresión, así como otros algoritmos de hash y cifrado pueden ser sencillamente añadidos a este protocolo.

Se realizó una implementación de DBRP que funciona como un proceso Daemon (*Disk And Execution MONitor*) responsable de administrar la información y tablas de enrutamiento entre los routers que tengan habilitado el protocolo. Adicionalmente, se desarrolló una herramienta gráfica (llamada *DBRP Settings*), responsable de generar el archivo de configuración que servirá de entrada al proceso Daemon para operar. El desarrollo de esta interfaz está basada en C++, haciendo uso de las bibliotecas de Qt. *DBRP Settings* permite (1) configurar la red y los parámetros del protocolo, siendo posible especificar direcciones y máscaras IPv4 e IPv6 con validación del correcto formato y posible solapamiento entre las interfaces configuradas, (2) manipular el estado de las interfaces, (3) asignar el retardo asociado a cada interfaz y a cada router, (4) crear y borrar interfaces virtuales, así como (5) activar y desactivar el protocolo para cada interfaz. También permite establecer mecanismos de seguridad en cuanto a autenticación y cifrado, visualizar las tablas de enrutamiento del sistema operativo, y configurar servicios comunes como direcciones de servidores DNS.

Finalmente fue desarrollado un complemento o plugin para la disección de los PDUs de DBRP. Mediante un árbol es posible desplegar los campos de los encabezados y de los TLVs, siendo posible detallar sus valores con una breve descripción.

Palabras Clave: Protocolo de Enrutamiento, DBRP, Seguridad, Propagación de Servicios Comunes, Qt.

Tabla de Contenido

Índice de Figuras

Índice de Tablas

1. Introducción

Los protocolos de enrutamiento se encargan de seleccionar las rutas por donde debe transitar la información en una red. Las decisiones son tomadas en base al cálculo de la mejor ruta; cada protocolo de enrutamiento selecciona la que considera su mejor ruta de acuerdo a la noción que tiene como métrica. La métrica no es más que la medida del costo asociado al recorrido de los datos del emisor al receptor en la red, la cual puede estar compuesta por una colección de redes más pequeñas llamadas ASs [6] (*Autonomous System*), cada una de las cuales empleará un determinado protocolo de enrutamiento para transmitir la información dentro del AS y hacia otros ASs.

Los protocolos que enrutan dentro de un AS son denominados IGPs [6] (*Interior Gateway Protocol*), mientras que los EGPs [6] (*Exterior Gateway Protocol*) son los que enrutan entre los distintos ASs. Es sumamente importante que los protocolos de tipo IGP funcionen de manera eficiente y sean capaces de responder ante los requerimientos de las redes actuales, lo cual implica un buen desempeño y estabilidad. El objetivo que nos motiva a realizar esta investigación será por tanto, analizar los protocolos IGPs existentes y enriquecer sus prestaciones al diseñar un nuevo protocolo que tome los mejores aspectos de cada uno de ellos y añada nuevas características que aseguren la robustez de su funcionamiento.

El diseño de un nuevo protocolo amerita un análisis detallado de la situación de los protocolos actuales para comprender sus limitaciones y dar cabida a nuevos enfoques que las resuelvan. Esta investigación está estructurada en cinco capítulos, los cuales son brevemente descritos a continuación:

Capítulo 2: El Problema.

Capítulo 3: Una breve revisión de los antecedentes de protocolos de enrutamiento de vector de distancia, destacando sus características generales y específicas así como sus limitaciones e impedimentos.

Capítulo 4: Se incursiona en algunos protocolos de enrutamiento de estado de enlace.

Capítulo 5: Descripción detallada del diseño estructural de DBRP, mostrando los tipos de mensajes y su funcionamiento.

Capítulo 6: Marco Metodológico.

Capítulo 7: Marco Aplicativo.

Capítulo 8: Especificaciones Técnicas de DBRP.

Capítulo 9: Conclusiones obtenidas producto de la investigación realizada y trabajos futuros.

2. El Problema

En este capítulo, se exponen los argumentos que justifican el diseño y desarrollo de un protocolo de enrutamiento adaptado a las nuevas necesidades de las redes actuales, y se plantean objetivos y alcances para incorporar una solución funcional.

2.1 Planteamiento del Problema

Los protocolos de enrutamiento de tipo IGP utilizados en la actualidad fueron diseñados hace aproximadamente dos décadas bajo requerimientos distintos a los de hoy en día; las redes actuales poseen cada vez más ancho de banda pero también mayor congestión, lo cual hace imprescindible tomar en cuenta con mucha prioridad parámetros como el retardo al momento de elegir las mejores rutas hacia un destino en la red. Adicionalmente, estos protocolos no abarcan una solución robusta a la vulnerabilidad que presentan las redes actuales ante ataques contra la seguridad de las mismas, en donde el cifrado de los datos de enrutamiento resulta en muchos casos sumamente necesario.

2.2 Justificación del Problema

El uso de redes informáticas ha venido incrementando la eficiencia y productividad de las sociedades, ocupando un papel fundamental en las empresas, institutos y organizaciones que se encuentran interconectadas a través de las mismas. Ello amerita que los protocolos de enrutamiento que rigen la comunicación de estas redes sean evaluados y mejorados constantemente, y que los mismos sean adaptados eficientemente a las necesidades que surgen en la actualidad.

2.3 Objetivos

A continuación se definen objetivos generales y específicos para la solución del problema.

2.3.1 Objetivos Generales

Diseñar e implementar un nuevo protocolo de enrutamiento de vector de distancia basado en el retardo capaz de transportar información de servicios comunes, que sea más seguro que los actuales al proveer autenticación y cifrado, y que se distribuya bajo la filosofía de código abierto.

2.3.2 Objetivos Específicos

- Hacer un análisis general de la forma en que trabajan los protocolos de enrutamiento de tipo IGP utilizados en la actualidad.
- Diseñar e implementar un protocolo de enrutamiento de vector de distancia que funcione a nivel de la capa de enlace del modelo de referencia OSI, en el

lenguaje de programación C++, haciendo uso de la comunicación entre procesos a nivel de sockets que provee la biblioteca sockets de C++.

- Desarrollar un mecanismo que permita el intercambio seguro de información de enrutamiento entre los routers involucrados en la comunicación, a través de la autenticación y del cifrado de los datos.
- Diseñar una interfaz gráfica basada en la biblioteca Qt [18] a través de la cual sea posible para los administradores configurar los parámetros del protocolo, ver la Tabla de Enrutamiento para IPv4 e IPv6, Tabla de Servicios Comunes para IPv4 e IPv6 y gestionar las interfaces de red.
- Realizar pruebas para validar el correcto funcionamiento del protocolo.

2.4 Alcances

DBRP es capaz de seleccionar la mejor ruta hacia un destino en la red basándose en el retardo. La solución ofrece soporte para redes IPv4 e IPv6, siendo posible extender el enrutamiento a otros protocolos de red, haciendo uso de la extensibilidad del protocolo a través de la definición de nuevos TLVs. Este protocolo no provee interoperabilidad con otros protocolos de enrutamiento de tipo IGP.

En la versión del protocolo a desarrollar dentro del contexto de este TEG es posible transportar únicamente servicios comunes como son las direcciones IP de servidores DNS.

Dado que DBRP está desarrollado en el lenguaje de programación C++ utilizando el compilador g++, las bibliotecas libg++ y libssl-dev, su implementación tiene lugar en PCs que ejecutan plataforma GNU/Linux basada en sistema de paquetes .deb.

La implementación a realizar debe contar con una interfaz de configuración gráfica que permita al administrador de red configurar los parámetros de DBRP de una manera sencilla. Adicionalmente permita la creación y eliminación de interfaces virtuales dummy siempre y cuando el respectivo módulo del kernel se encuentre instalado en la distribución.

3. Antecedentes de Protocolos de Enrutamiento de Vector de Distancia

Desde sus comienzos Internet está organizado en un gran número de redes interconectadas a través de routers, estas redes pueden estar basadas en conexiones punto a punto o redes más complejas como Ethernet. Para lograr este esquema, entran en juego los protocolos de enrutamiento que permiten a un router decidir por dónde enviar un datagrama IP ya sea entregándolo inmediatamente a su destino si éste está directamente conectado al router o a la puerta de enlace más cercana (*Next Hop*) cuando el destino pertenece a otra red que no es directamente alcanzable. En otras palabras, el objetivo de un protocolo de enrutamiento es proveer la información necesaria para alcanzar el destino.

Uno de los primeros protocolos de enrutamiento usados en la década de los ochenta fue RIPv1 [12] (*Routing Information Protocol*), el cual es un estándar abierto. Para aquella época las redes eran más pequeñas y menos congestionadas. Este protocolo se basa en la distancia existente entre dos hosts, la cual está determinada por el número de routers intermedios por los que debe pasar un datagrama IP para alcanzar su destino. Este tipo de métrica es conocida como Vector de Distancia y utiliza el algoritmo de Bellman-Ford [5] para calcular las rutas, éste fue el algoritmo original utilizado por ARPANET.

Otro protocolo de Vector de Distancia ampliamente utilizado es IGRP [11] (*Interior Gateway Routing Protocol*) que a diferencia de RIP, es una tecnología propietaria de Cisco Systems, que surge con el objetivo de mejorar la elección de una determinada ruta tomando en cuenta elementos que no contemplaba RIP.

3.1 RIP

En redes grandes, como Internet, es poco probable que un único protocolo de enrutamiento sea usado para toda la red. En vez de esto, dicha red de redes será organizada como una colección de ASs (*Autonomous Systems*) cada uno de los cuales será administrado por una sola entidad.

Cada AS tendrá su propia tecnología de enrutamiento, la cual puede diferir de otros ASs. Un protocolo de enrutamiento usado dentro de un AS es conocido como IGP (*Interior Gateway Protocol*). Por otro lado, un EGP (*Exterior Gateway Protocol*) es el protocolo destinado al intercambio de información de enrutamiento entre dos o más ASs. RIPv1 fue uno de los primeros protocolos de enrutamiento usado en redes TCP/IP, cuya implementación estuvo disponible antes de que la especificación oficial fuese publicada, esto fue posible gracias a que fue incluido en el sistema operativo BSD [17] (Berkeley Software Distribution) en la década de los ochenta.

3.1.1 Descripción General del Protocolo

La tarea de encontrar una ruta desde un emisor hasta un destino deseado es a lo que se llama enrutamiento, y esto en IP [12] (*Internet Protocol*) básicamente se reduce a encontrar las puertas de enlace que comunican con esos destinos. Dentro de un mismo dominio de *broadcast* cualquier transferencia de paquetes será resuelta por la tecnología empleada en esa misma red, el enrutamiento como tal se presentará cuando un emisor decida comunicarse con un destino que se encuentra en una red diferente, para lo cual será necesaria la intervención de uno o más routers.

RIP ofrece una manera de encontrar rutas entre redes interconectadas, a través del intercambio de información entre routers. Cada entidad involucrada en el intercambio mantiene información en una tabla de enrutamiento que contiene todos los destinos que son alcanzables por dicha entidad. En general todas las entidades pertenecientes a un mismo dominio de *broadcast* serán vistas como una única red y se establecerá un camino hacia ella a través de una sola entrada en la tabla de enrutamiento.

Para que un router tome la decisión de enviar un mensaje hacia determinada red destino, es necesario mantener cierta información provista por los routers adyacentes o vecinos. Esta información es compartida periódicamente (cada 30 segundos) entre los routers, lo cual les permitirá enrutar adecuadamente los mensajes. En la Figura 3.1, se puede observar una red donde se utiliza RIPv1 como IGP. Cuando un host perteneciente a la red 192.168.4.0/24 desea comunicarse con otro host de la red 192.168.1.0/24, debe enviar los paquetes hacia su puerta de enlace, que en este caso es el router R4. R4 deberá reenviar esos paquetes a R2 y éste último hacia R1, ya que como puede verse en la Figura 3.2, la tabla de enrutamiento de R4 así lo indica con métrica de 2 saltos.

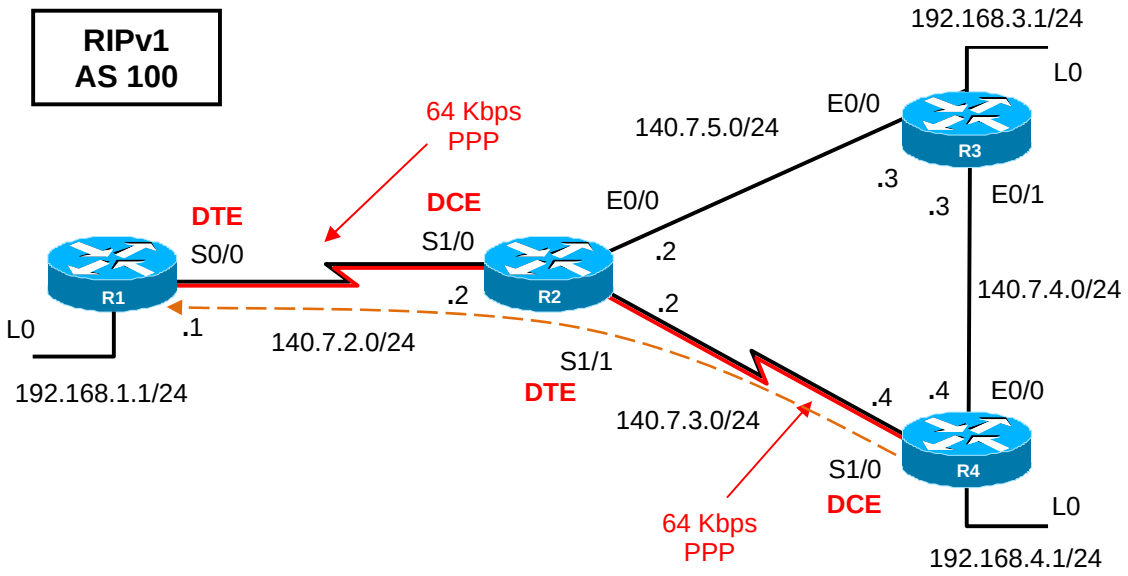

Figura 3.1: Ejemplo de una Red que Utiliza RIPv1

```
R4# show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       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
       U - per-user static route

C   192.168.4.0 is directly connected, Loopback 0
    140.7.0.0/24 is subnetted, 4 subnets
C   140.7.3.0 is directly connected, Serial1/0
C   140.7.4.0 is directly connected, Ethernet0/0
R   140.7.2.0 [120/1] via 140.7.3.2, 00:07:41, Serial1/0
R   140.7.5.0 [120/1] via 140.7.3.2, 00:03:40, Serial1/0
R   192.168.1.0 [120/2] via 140.7.3.2, 00:02:41, Serial1/0
R   192.168.3.0 [120/1] via 140.7.4.3, 00:01:17, Ethernet0/0
```

Figura 3.2: Tabla de Enrutamiento de R4 para RIPv1

3.1.2 Descripción Específica del Protocolo

Todo dispositivo que usa RIP debe mantener una tabla de enrutamiento con al menos la siguiente información:

- **RIPv1**
  - Identificador de la red destino.
  - Métrica: la cual representa el costo total para enviar un datagrama desde un router a ese destino, siendo ésta el número de routers intermedios.
  - Dirección IPv4 del siguiente router en la ruta: en caso de que el destino no esté directamente conectado.
  - Una bandera para indicar que la información de la subred destino ha sido modificada recientemente.
  - Varios indicadores de tiempo asociados a la subred destino.

Para redes directamente conectadas la métrica será de valor 0 ya que no hay routers intermedios. Algunas implementaciones pueden usar métricas más complejas cuando se desea mostrar preferencia a una determinada red sobre otra, por ejemplo si se toma en cuenta el ancho de banda o la confiabilidad. Debido a la limitación de 15 saltos que establece la métrica de RIP, algunas implementaciones permiten incluir manualmente rutas adicionales que por lo general pueden ser un determinado host o una red destino que está fuera del alcance del protocolo de enrutamiento.

Cada 30 segundos debe ser enviado un mensaje de actualización por todas las interfaces RIP, conteniendo la información descrita en la tabla de enrutamiento. Para evitar colisiones con otros routers al momento del envío, se resta una fracción de

tiempo aleatorio al valor de 30 segundos, de modo tal que no todos los routers harán su anuncio al mismo tiempo.

Cuando se presenta un cambio en la topología de la red, se debe esperar al siguiente mensaje de actualización síncrono para que los routers vecinos sean notificados. Este tiempo puede ser reducido con el uso de *Triggered Updates*, que consiste en enviar de manera asíncrona, un mensaje de actualización antes de los 30 segundos, con sólo aquellas entradas de las rutas que han sido modificadas.

Un problema que se presenta en estos sistemas es el llamado Ciclo de Enrutamiento, donde un router A contiene en su tabla una entrada para determinada ruta indicando que el router B es su *Next Hop*, a su vez el router B contiene esa misma ruta pero alcanzable a través de A. Esto causa un ciclo indeseado, ya que los datagramas quedan encerrados entre ambos routers. Este problema se puede resolver utilizando *Split Horizon*, el cual indica que las subredes no deben ser anunciadas por la misma interfaz por la que fueron aprendidas.

La Figura 3.3 ilustra este esquema, por ejemplo el router R1 inicialmente anuncia la Red A, a continuación el router R2 incluirá esta misma red en la actualización que le enviará posteriormente a R1. *Split Horizon* indica que en esa actualización de R2 hacia R1, la Red A no debe ser anunciada. Obsérvese la siguiente situación, si la interfaz de R1 directamente conectada a la Red A falla, R2 continuará informando a R1 que puede alcanzar a la Red A (a través de R2). Si R1 confía en este anuncio, y agrega esta red en su tabla, ello sería un error, ya que ninguno de ellos podrá alcanzar la Red A, causando un ciclo de enrutamiento.

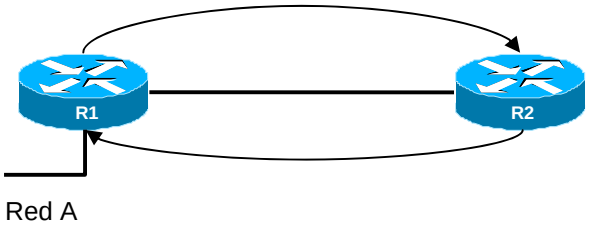

Figura 3.3: Esquema Split Horizon

Adicionalmente, se puede usar este esquema (*Split Horizon*) en combinación con *Poisson Reverse*, el cual sugiere que se anuncien todas las entradas de la tabla pero aquellas rutas que fueron aprendidas por una interfaz sean anunciadas con métrica de valor 16 por esa misma interfaz. De este modo el router que recibe el anuncio, ignorará esas rutas que son explícitamente marcadas como inalcanzables, fortaleciendo la prevención de un ciclo de enrutamiento.

Los dos indicadores de tiempo, *Timeout* y *Garbage-collector* [12] de cada prefijo en la tabla, controlan diferentes eventos. El primero se inicializa con valor 0 cuando se establece la ruta por primera vez o cuando es recibido un mensaje de actualización para esa ruta, pasados 180 segundos desde la última inicialización, se considera que la ruta ha expirado. El segundo indicador es inicializado con valor 0 cuando se considera

que la ruta ha expirado o cuando es recibido un mensaje de actualización indicando que la ruta es inalcanzable, en ambos casos la métrica para dicha ruta se establece en 16 (Infinito), y la ruta es eliminada de la tabla una vez transcurridos 120 segundos del indicador *Garbage-collector*.

- **RIPv2:** a diferencia de RIPv1, RIPv2 [15] es capaz de manejar información de las subredes, incluyendo en su formato de mensaje un campo para la máscara de subred (*Subnet Mask)*. La comunicación es *multicast* y autenticada.

- **RIPng:** a diferencia de RIPv1, RIPng [16] está diseñado para redes basadas en IPv6, adaptando el formato de sus mensajes únicamente para soportar direcciones IPv6.

3.1.3 Desventajas e Impedimentos del Protocolo

Este protocolo, en cualquiera de sus versiones (RIPv1, RIPv2 y RIPng), no está diseñado para redes de gran tamaño, de hecho está limitado a una métrica de 15 saltos como máximo, considerando como inalcanzable cualquier valor mayor a este.

En RIP*, Counting to Infinity* es uno de los problemas que se presenta cuando uno o varios routers lógicamente ya no pueden alcanzar un destino debido a un ciclo de enrutamiento, y en cada actualización la métrica asociada a la red destino se incrementará hasta alcanzar un valor considerado como inalcanzable. Esta situación puede ocasionar una lenta convergencia, lo cual afecta notablemente el rendimiento de la red. Para reducir este tiempo de convergencia se implementan mecanismos como *Split Horizon* y *Poisson Reverse* descritos anteriormente en la Subsección 3.1.2.

La manera como este protocolo define la métrica, no resulta apropiada para situaciones donde las rutas son seleccionadas en base a parámetros de tiempo real, tales como retardo (*delay*), ancho de banda (*bandwidth*), confiabilidad (*reliability*) y carga (*load*). Las redes actuales demandan cada vez más velocidad y eficiencia en la elección de las rutas, por lo cual surgieron protocolos alternativos como IGRP, el cual será descrito más adelante.

3.2 IGRP

IGRP [11] (*Interior Gateway Routing Protocol*) es un IGP diseñado a mediados de los años ochenta por Cisco Systems. Fue desarrollado principalmente para superar la limitación que presenta RIPv1 con respecto a su métrica de 15 saltos como máximo. IGRP tal como otros protocolos de tipo IGP está orientado a un conjunto de redes sencillas, sin embargo puede escalar redes más grandes y complejas.

3.2.1 Descripción General del Protocolo

IGRP se basa en el intercambio de información entre routers, manteniendo también una tabla de enrutamiento. Esto lo hace mediante mensajes de actualización que recibe de los routers vecinos regularmente cada 90 segundos. A medida que estos

mensajes son propagados, los routers pueden calcular la distancia hacia cualquier nodo de la red; esto no quiere decir que un router conoce toda la topología, sino que sólo sabe lo que aprende de sus vecinos, a diferencia de los protocolos de estado de enlace donde sí se conoce la totalidad de la topología de la red [6].

A diferencia de RIP en cualquiera de sus versiones, IGRP no utiliza un único valor para su métrica, sino que es una composición de varios factores determinantes en la escogencia de la mejor ruta, como lo son: retardo (*delay*), ancho de banda (*bandwidth*), confiabilidad (*reliability*) y carga (*load*). Todos estos valores son unificados en un solo valor mediante una fórmula.

Obsérvese en la Figura 3.4 la misma topología mostrada en la Figura 3.1, pero ahora utilizando el protocolo de enrutamiento IGRP. El mismo host perteneciente a la red 192.168.4.0/24 que desea comunicarse con el host de la red 192.168.1.0/24 debe enviar los paquetes hacia su puerta de enlace que en este caso es el router R4, éste deberá reenviar esos paquetes a través del router R3 ya que la tabla de enrutamiento de R4 así lo indica, como puede verse en la Figura 3.5. Anteriormente RIPv1 lo enviaba a través de R2 ya que está a una menor cantidad de saltos, pero ahora IGRP toma en cuenta el ancho de banda y el retardo, y por tanto decide tomar como próximo salto a R3 ya que los enlaces Ethernet son de mayor velocidad que los seriales a pesar de que está a un mayor número de saltos.

Adicionalmente IGRP permite que el administrador modifique este valor a su conveniencia cuando por ejemplo desee dar prioridad a una ruta en particular o cuando desee dar mayor importancia a la confiabilidad que al ancho de banda, por tanto el protocolo resulta bastante flexible.

En líneas generales IGRP tiene los siguientes objetivos [11]:

- Prevenir los ciclos de enrutamiento, por lo que provee estabilidad inclusive en redes grandes y complejas.
- Responder rápidamente a cambios de topología de la red.
- Hacer balanceo de carga por rutas con diferentes métricas.
- Tomar en cuenta la tasa de error y los niveles de tráfico de las diferentes rutas.

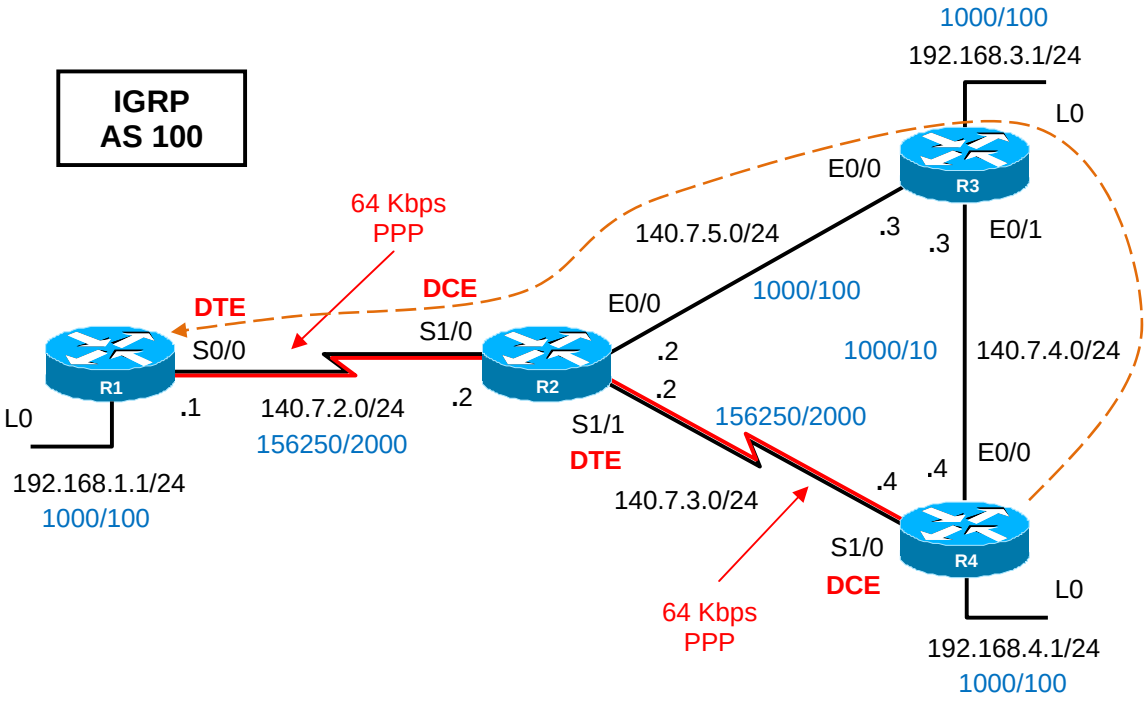

Figura 3.4: Topología Usando IGRP

```
R4#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       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
       U - per-user static route

C    192.168.4.0 is directly connected, Loopback 0
     140.7.0.0/24 is subnetted, 4 subnets
C    140.7.3.0 is directly connected, Serial1/0
C    140.7.4.0 is directly connected, Ethernet0/0
I    140.7.5.0 [100/1200] via 140.7.4.3, 00:03:33, Ethernet0/0
I    140.7.2.0 [100/158450] via 140.7.4.3, 00:04:36, Ethernet0/0
I    192.168.3.0 [100/1200] via 140.7.4.3, 00:04:33, Ethernet0/0
I    192.168.1.0 [100/158550] via 140.7.4.3, 00:08:19, Ethernet0/0
```

Figura 3.5: Tabla de Enrutamiento de R4 para IGRP

3.2.2 Descripción Específica del Protocolo

En 1994 Cisco Systems publica una versión mejorada de IGRP con el objetivo de manipular redes de alto crecimiento y misión crítica. Esta nueva versión es conocida como EIGRP (*Enhanced Interior Gateway Routing Protocol*) la cual mejora las propiedades de convergencia y opera con mayor eficiencia que IGRP. A continuación se describen las características de ambas versiones.

- **IGRP:** usa directamente la capa IP, diferenciando los sistemas autónomos que lo componen a través de números entre los distintos dominios de enrutamiento. Es un protocolo *classful*; es decir que no maneja máscara de subred ya que se aplica la máscara de acuerdo a la máscara por defecto de la dirección recibida.

  Cuando un router recibe un anuncio de un prefijo con mayor métrica o inalcanzable, es puesto en *holddown*, que consiste en ignorar durante un período de tiempo los anuncios de dicho prefijo. Si durante este tiempo el prefijo es anunciado por otro router que no fue por donde se aprendió inicialmente, el anuncio será ignorado. Este mecanismo suele ser usado para evitar los ciclos de enrutamiento, pero tiene el efecto de incrementar el tiempo de convergencia[1]. IGRP también hace uso de *Split Horizon* y *Poisson Reverse* mencionados en la Subsección 3.1.2.

  Este protocolo establece los siguientes rangos de los valores que componen la métrica. Confiabilidad y carga pueden tomar valores de 1 a 255, ancho de banda puede tomar valores que reflejen velocidades desde 1.2 Kbps hasta 10 Gbps, mientras que el retardo puede tomar cualquier valor entre 1 y 2^{24}; esto permite tomar en cuenta con precisión las capacidades de un determinado enlace para la elección de una métrica más justa.

  IGRP mantiene un conjunto de variables de tiempo que incluyen, el tiempo de actualización (*Update Timer*) indicando el intervalo de tiempo en el cual debe ser enviada una actualización (por defecto 90 segundos). El tiempo de expiración (*Invalid Timer*) indica cuánto tiempo debe esperar un router sin recibir actualizaciones para considerar una ruta inválida, por defecto este valor es tres veces el tiempo de actualización. El tiempo de espera (*Holddown Timer*) especifica el tiempo que se deben ignorar los anuncios de un determinado prefijo, por defecto este valor es tres veces el tiempo de actualización más 10 segundos. Finalmente el tiempo de eliminación (*Flush Timer*) indica el tiempo en que debe ser eliminada una entrada de la tabla de enrutamiento, por defecto este tiempo es de siete veces el tiempo de actualización [6].

[1] http://www.cisco.com/en/US/docs/ios/12_0/np1/configuration/guide/1cigrp.html#wp4768

- **EIGRP:** Los routers que manejan EIGRP [22] mantienen información de enrutamiento y de la topología en la memoria RAM, de manera que son capaces de reaccionar rápidamente ante los cambios. El algoritmo DUAL [5] (*Diffusing Update Algorithm*) es el algoritmo utilizado por EIGRP, el cual permite que los routers involucrados se comuniquen de manera eficiente y que en un cambio de topología se sincronicen al mismo tiempo sin involucrar a los routers que no se ven afectados por el cambio, este algoritmo evita los ciclos de enrutamiento.

  EIGRP genera menos tráfico que IGRP, porque es capaz de limitar el intercambio de información de enrutamiento para incluir solamente la información que ha cambiado. El envío y recepción de los paquetes EIGRP de forma confiable es hecho con RTP (*Reliable Transfer Protocol*) quien garantiza que los paquetes sean entregados en orden.

  En EIGRP los paquetes pueden ser autenticados usando una suma de comprobación con criptografía MD5 (*Message Digest 5*). De esta manera se previene que fuentes no confiables introduzcan paquetes no autorizados o mensajes de enrutamiento falsos [22].

  EIGRP mantiene las siguientes tablas:

  - Tabla de vecinos.
  - Tabla de enrutamiento.
  - Tabla topológica.
  - Tabla topológica completa.

  Cada router mantiene una tabla de vecinos donde se encuentra la información de los routers adyacentes. Existe una tabla de vecinos por cada protocolo que admite EIGRP, tales como AppleTalk, IPX e IP.

  Al establecer la adyacencia con un nuevo vecino, se registra la dirección IP del mismo y la interfaz por la cual se estableció la comunicación. Cuando un vecino envía un paquete *Hello*, anuncia el *Holddown Timer*, estableciendo el período durante el cual un router considera que un vecino se puede alcanzar y que sigue activo. Si uno de estos paquetes no se recibe dentro del tiempo esperado, el vecino será considerado inalcanzable, por tanto debe ser recalculada la tabla de vecinos así como las rutas que eran alcanzables a través de él.

  La tabla topológica se compone de todas las tablas de enrutamiento EIGRP en el sistema autónomo. El algoritmo DUAL toma la información proporcionada en la tabla de vecinos y la tabla topológica para calcular las rutas de menor costo hacia cada destino. EIGRP rastrea esta información para que los routers puedan identificar y conmutar a rutas alternativas rápidamente. La información que el router recibe de DUAL se utiliza para determinar la ruta del sucesor, término utilizado para identificar la ruta principal o la mejor. Pueden existir hasta cuatro rutas de sucesor para cada destino en particular, las cuales pueden ser de costo

igual o desigual y se identifican como las mejores rutas sin ciclos hacia un destino determinado. Los routers mantienen una tabla topológica por cada protocolo de red configurado.

El uso del algoritmo DUAL hace que EIGRP sea libre de ciclos de enrutamiento, valiéndose del cálculo distribuido de las rutas en cada nodo involucrado. Éste cálculo sólo toma en cuenta las rutas directamente conectadas en cada nodo y posteriormente el resultado del cálculo es compartido con los routers adyacentes, haciendo un uso eficiente del ancho de banda.

Es posible registrar información adicional acerca de cada prefijo en la tabla topológica. EIGRP los clasifica como internos o externos, agregando un rótulo a cada uno para identificar esta clasificación. Los prefijos internos se originan dentro del AS de EIGRP, mientras que los externos se originan fuera. Los prefijos aprendidos desde otros protocolos de enrutamiento como RIP, OSPF o IGRP son considerados externos.

La tabla de enrutamiento mantiene las rutas que se aprenden de forma dinámica, y contiene las mejores rutas hacia un prefijo. Esta información se recupera de la tabla topológica. Se mantiene una tabla de enrutamiento por cada protocolo de red que maneja.

- **EIGRPv6:** Debido al diseño extensible de IGRP y EIGRP con el uso de TLVs (*Type Length Value*) para incluir nuevas características, fue fácil extender EIGRP a IPv6. EIGRPv6 (versión de EIGRP para IPv6) define nuevos TLVs para dar soporte a IPv6, el comportamiento y las características de EIGRP se conservan.

- **Easy-EIGRP** [8]**:** esta es una herramienta desarrollada por estudiantes de la Escuela de Computación de la Universidad Central de Venezuela, orientada al aprendizaje del protocolo. Es una implementación del protocolo EIGRP desarrollada en Java. Su objetivo es el apoyo didáctico para la enseñanza en cursos avanzados de redes. La principal característica de Easy-EIGRP es que provee una interfaz gráfica (ver Figura 3.6) intuitiva que permite configurar los aspectos más relevantes de la red y del protocolo en sí. Implementa los siguientes módulos, visor parcial de la red, registro de eventos, un sniffer, visor de las tablas del protocolo (tabla de vecinos, tabla de enrutamiento, tabla topológica y tabla topológica completa) y una representación gráfica de la máquina de estado finito DUAL.

  El desarrollo de esta aplicación fue basado en el estudio de situaciones puntuales aplicando ingeniería inversa para comprender el comportamiento de EIGRP debido a la escasez de documentación. Su diseño permite su instalación en cualquier computador configurado como router en plataformas Windows/Linux gracias a la portabilidad que ofrece Java.

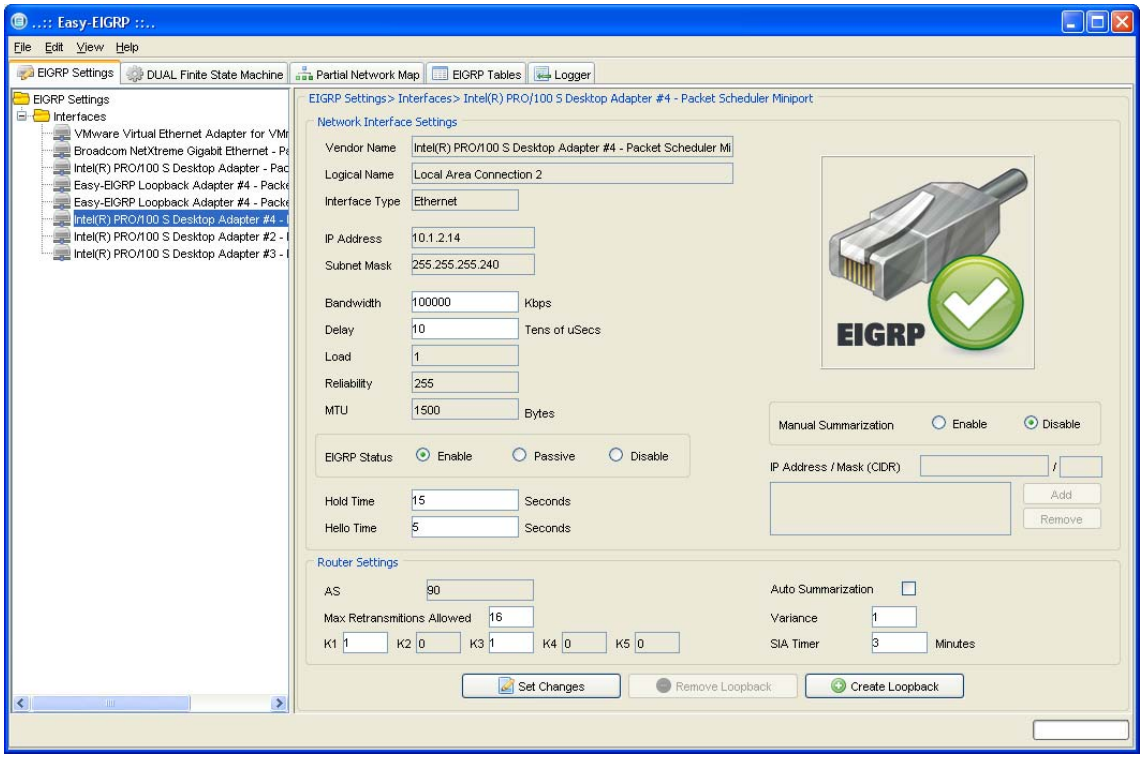

Figura 3.6: Ejemplo de Interfaz Gráfica de Configuración de Easy-EIGRP

3.2.3 Desventajas e Impedimentos del Protocolo

Dado que estos protocolos poseen un tipo de licenciamiento propietario, no es posible utilizar o implementar alguno de éstos en dispositivos que no sean fabricados por Cisco.

IGRP es un protocolo *classful*, lo que significa que no transporta la máscara de red y se asume la máscara de la interfaz de recepción. Adicionalmente es propenso a ciclos de enrutamiento, lo que eventualmente produce una lenta convergencia.

IGRP y EIGRP envían muchos mensajes para su funcionamiento. IGRP envía mensajes periódicos con la totalidad de la tabla de enrutamiento. Esto último no ocurre en EIGRP, dado que está programado para no consumir más del 50% de la capacidad del enlace en el envío de los mensajes [7].

SIA (*Stuck In Active*) es una de las desventajas que presenta EIGRP. Ocurre cuando un router no ha recibido un mensaje de respuesta a una consulta que realizó a sus vecinos. EIGRP envía un mensaje de consulta cuando se ha perdido el camino a un prefijo dado.

En la Tabla 3.1 (modificada de [17]), se puede observar una comparación entre las características principales de los protocolos de enrutamiento descritos en el Capítulo 3.

Protocolo	RIPv1	RIPv2	RIPng	IGRP	EIGRP	EIGRPv6
Familia de Direcciones	IPv4	IPv4	IPv6	IPv4	IPv4	IPv6
Métrica	Salto	Salto	Salto	Compuesto	Compuesto	Compuesto
Comunicación	Broadcast	Multicast	Multicast	Broadcast	Multicast	Multicast
Cálculo de Ruta	Bellman-Ford	Bellman-Ford	Bellman-Ford	Bellman-Ford	DUAL	DUAL
VLSM/ CIDR	No	Sí	Enmascara-miento IPv6	No	Sí	Enmascara-miento IPv6
Observación	Lenta convergen-cia, Split Horizon	Lenta convergen-cia, Split Horizon	Lenta convergen-cia, Split Horizon	Lenta convergen-cia, Split Horizon	Convergen-cia rápida y libre de ciclos, exceso de mensajes	Convergen-cia rápida y libre de ciclos, exceso de mensajes

Tabla 3.1: Comparación de Protocolos en la Familia de Vector de Distancia

4. Protocolos de Enrutamiento de Estado de Enlace

Protocolos como OSFP e IS-IS registran el estado y el tipo de conexión de cada enlace para producir una métrica basada principalmente en estos factores. Estos protocolos sólo envían la información requerida cuando hay cambios de topología en la red.

4.1 OSPF

OSPF (*Open Shortest Path First*) es un IGP desarrollado por IETF (*Internet Engineering Task Force*) orientado a redes IP. Fue creado a raíz de que RIP comenzó a ser inapropiado para redes grandes y heterogéneas. Este protocolo tiene dos principales características; la primera de ellas es que el protocolo es abierto y su especificación es de dominio público en el RFC 1131 [19] para OSPFv1 y en el RFC 2328 [20] para OSPFv2. La segunda y más importante, es que OSPF está basado en el algoritmo de Dijkstra [17], el cual es un excelente método para encontrar el camino más corto hacia un prefijo dado.

4.1.1 Descripción General del Protocolo

OSPF envía LSAs (*Link State Advertisement*) a los demás routers dentro de la misma área de jerarquía. En estos LSAs, se incluye información como números de enlaces, tipos, costos, identificador de subred, máscara de subred, etc. Dado que cada router OSPF acumula información del estado del enlace, ellos usan el algoritmo SPF [17] (*Shortest Path First*) para calcular la ruta más corta hacia cada nodo.

En redes BMA (*Broadcast Multi-Access*), donde hay varios routers, el intercambio de mensajes *Hello* elige un router designado o DR (*Designated Router*) y un router designado de respaldo o BDR (*Backup Designated Router*). El router designado es responsable de retransmitir los LSAs, lo cual permite una reducción en el tráfico de la red [20].

4.1.2 Descripción Específica del Protocolo

Cuando un router es inicializado usa mensajes *Hello* de OSPF para descubrir a sus vecinos. Este intercambio de mensajes *Hello* también es usado para asegurarse que los routers vecinos siguen funcionando. A través de la comparación de adyacencias establecidas y el estado del enlace, un router que falla puede ser detectado rápidamente.

Un internet puede ser dividido en un conjunto de áreas, las cuales son agrupadas en redes contiguas. Los routers con múltiples interfaces pueden participar en múltiples áreas, éstos son llamados ABRs (*Area Border Routers*), los cuales mantienen bases de datos topológicas separadas para cada área.

Una base de datos topológica es en esencia, una vista global de la red donde el camino más corto hacia un prefijo es indicado por el SPT (*Shortest-Path Tree*). Contiene la colección de los LSAs recibidos de todos los routers de la misma área más unos resúmenes de las otras áreas. Los routers de una misma área comparten la misma información y por tanto poseen una base de datos idéntica.

La topología en detalle de un área es invisible a las entidades fuera de la misma, manteniendo las topologías separadas. En consecuencia, OSPF genera menos tráfico para el intercambio de LSAs del que generaría si los internets no estuviesen particionados. El particionamiento de áreas crea dos diferentes tipos de enrutamiento OSPF dependiendo de si el origen y el destino se encuentran en la misma o en diferentes áreas. El enrutamiento intra-área ocurre cuando el origen y el destino se encuentran en la misma área, mientras que el enrutamiento inter-área ocurre cuando están en diferentes áreas.

El *backbone* de OSPF es responsable de distribuir la información de enrutamiento entre las diferentes áreas. Es el área 0 donde están los ABRs y algunos routers internos.

La Figura 4.1 muestra un ejemplo de una red con múltiples áreas, los routers R3, R4, R5, R8, R9 y R10 conforman el *backbone* de OSPF. Si el host H1 en el Área 3, desea comunicarse con el host H2 en el Área 2, los paquetes son enviados al router R12, el cual reenviará dichos paquetes a R10, R9 y R8 a través del *backbone*, luego a través del router intra-área R6, para ser finalmente entregados al host H2.

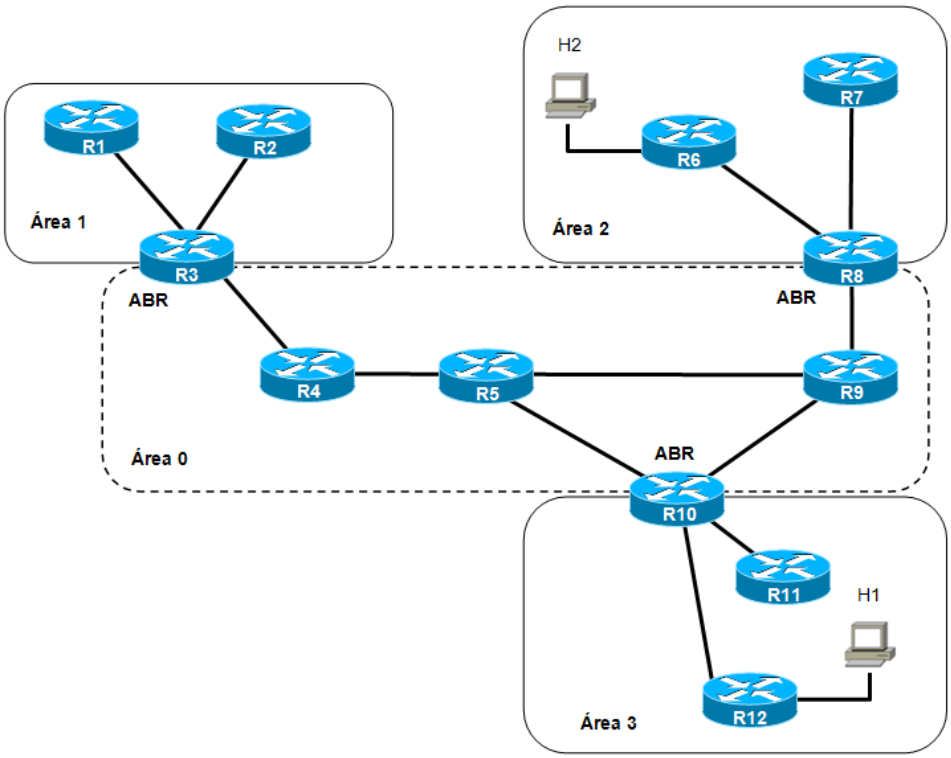

Figura 4.1: Red OSPF con Múltiples Áreas

Las áreas pueden ser definidas de tal manera que el *backbone* no sea contiguo, en este caso la conectividad del *backbone* debe ser restablecida a través de enlaces virtuales. Estos pueden ser configurados entre cualquiera de los routers del *backbone* que compartan un enlace hacia un área fuera del mismo, de esta manera funciona como si estuvieran directamente conectados [6].

4.2 IS-IS

IS-IS (*Intermediate System to Intermediate System*) fue desarrollado por DEC (*Digital Equipment Corporation*) a finales de los años ochenta, es un IGP de estado de enlace que ofrece jerarquía a través de áreas. Usa el algoritmo de Dijkstra para calcular la mejor ruta hacia un prefijo dado.

El protocolo fue definido en ISO/IEC 10589:2002 como un estándar internacional dentro del modelo de referencia OSI (*Open Systems Interconnection*) con el objetivo de enrutar datagramas usando la pila de protocolos CLNS (*Connectionless Network Service*). Posteriormente fue publicado por la IETF en el RFC 1142 [21]; la IETF fue responsable de importantes cambios en IS-IS, uno de ellos fue la extensión del protocolo al definir nuevos TLVs portando nuevas funcionalidades. También, se aclararon muchos aspectos operacionales del protocolo en el RFC 1195 [2]; por ejemplo, el manejo de las adyacencias que no había sido del todo definido.

4.2.1 Descripción General del Protocolo

IS-IS y OSPF son protocolos conceptualmente similares, soportan VLSM, emplean comunicación *multicast* para descubrir routers vecinos a través de mensajes *Hello* y adicionalmente usan autenticación en los mensajes de actualización.

Los routers de IS-IS construyen una representación de la topología de la red, indicando las subredes IP que cada router puede alcanzar y el costo más bajo hacia las mismas. IS-IS requiere una cantidad considerablemente menor de mensajes que OSPF para su funcionamiento, además de la capacidad de escalar en redes de mayor tamaño, lo cual ha sido comprobado debido al uso que le han dado muchos ISPs (*Internet Service Provider*).

4.2.2 Descripción Específica del Protocolo

En sus comienzos los protocolos de estado de enlace consistían en un conjunto de routers que compartían una base de datos topológica común para calcular la mejor ruta hacia un destino. En ese entonces los investigadores en su proceso de estandarización ponían en duda la escalabilidad del algoritmo SPF y de dichos protocolos de enrutamiento en general. La idea para una buena escalabilidad fue dividir una red grande en porciones disjuntas más pequeñas llamadas Áreas, con el objetivo de reducir sustancialmente el costo de cómputo en cada router a nivel de procesamiento. Para poder comunicar estas áreas, es necesario designar ciertos routers para el intercambio de información entre las mismas.

La Figura 4.2 ejemplifica esta situación en la que se divide el Área 49.0005 en dos Áreas (49.0002 y 49.0008), el router R3 antes anunciaba a R4 las redes 172.16.1.0/24, 172.16.2.0/24 y 172.16.3.0/24, y R4 de igual manera anunciaba a R3 las redes 172.17.1.0/24, 172.17.2.0/24 y 172.17.3.0/24. Con la división, ahora las redes que anuncian R3 y R4 se sumarizan, reduciendo el número de prefijos en los mensajes, por tanto R3 ahora anuncia a R4 el prefijo 172.16.0.0/22 y R4 anuncia a R3 el prefijo 172.17.0.0/22 [9].

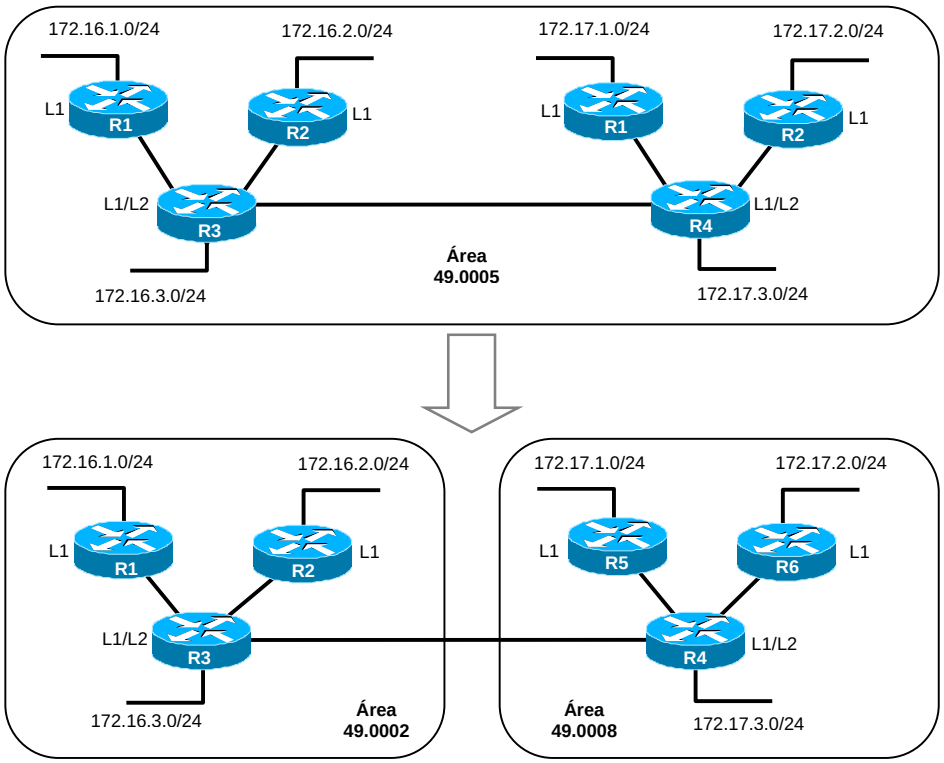

Figura 4.2: Área en IS-IS

Otro concepto importante que introduce IS-IS son los llamados niveles de enrutamiento, que definen de una manera jerárquica como se deben comunicar los routers. Se tienen dos niveles de jerarquía, los routers de Nivel 2 (L2) pasan a conformar lo que en OSPF se entiende como *backbone*, con la diferencia de que no es identificado como Área 0, estos routers se encargan del intercambio de tráfico entre áreas.

Los routers de Nivel 1 (L1) se encargan de manejar el tráfico dentro de su área, por tanto la base de datos topológica es común para los routers que pertenecen a una misma área. También pueden configurarse routers L1/L2 que son capaces de cumplir ambas tareas. Para comunicarse con otra área, los routers L1 deben reenviar los paquetes al router L1/L2 más cercano en caso de que existan varios.

La Figura 4.3 provee un ejemplo de cómo se comunican routers que se encuentran en distintas áreas, cada área posee al menos un router L1/L2 o L2. Al nivel 2, un router puede tener vecinos tanto en la misma como en otras áreas y su base de datos contendrá información de enrutamiento inter-área pero no posee la misma información que maneja un router L1, ya que éste último sólo conoce la información de enrutamiento intra-área. Los routers L1/L2 mantienen bases de datos separadas para cada nivel[2].

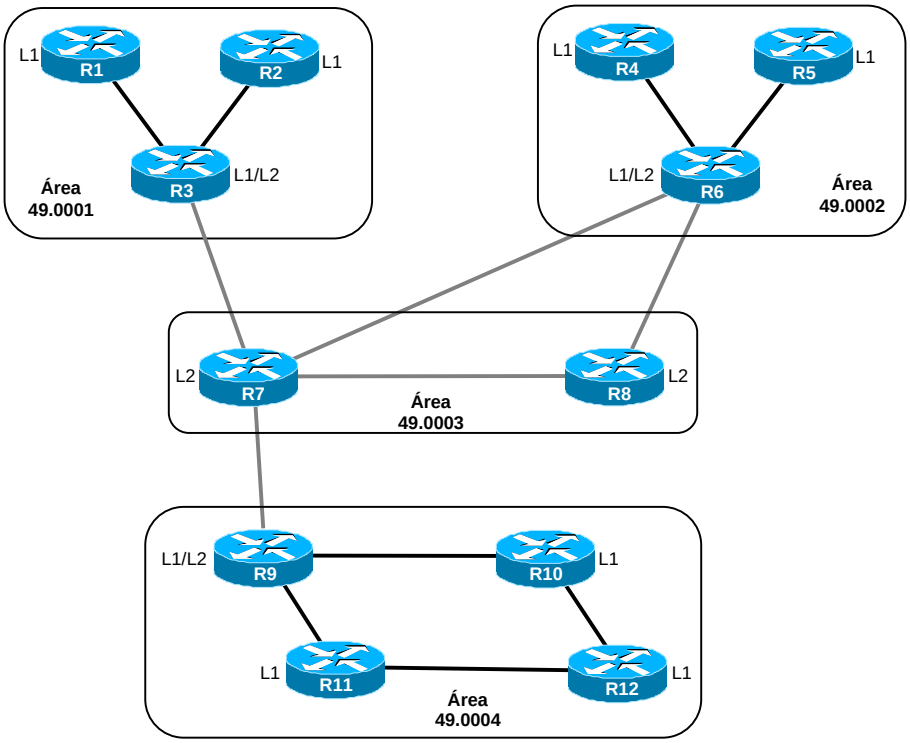

Figura 4.3: Jerarquía de Niveles en IS-IS

[2] http://www.cisco.com/en/US/products/ps6599/products_white_paper09186a00800a3e6f.shtml

5. Protocolo de Enrutamiento de Vector de Distancia Basado en el Retardo

DBRP (*Delay Based Routing Protocol*) es un protocolo de enrutamiento de Vector de Distancia cuya métrica está basada en el retardo. El intercambio de información de enrutamiento se hace mediante anuncios periódicos, incorporando mecanismos de seguridad tanto de autenticación como de cifrado, siendo desarrollado bajo la filosofía de código abierto. A lo largo de este capítulo se describirán los fundamentos técnicos de su diseño.

5.1 Estructura del Protocolo

DBRP posee varios tipos de mensajes que definen la manera en que los routers se comunican entre sí, diseminando información de enrutamiento que permitirá la elección de rutas eficientes entre los nodos de la red. Estos mensajes son:

- *Update*: mensaje que envía un router para anunciar información de enrutamiento. El envío de este mensaje es periódico y está determinado por el *Update Interval* que por defecto es de 30 segundos. Adicionalmente los *Updates* permiten enviar información de servicios comunes a los routers para simplificar la administración de los servicios.
- *Request*: permite a un router solicitar a sus vecinos información de enrutamiento y de servicios comunes.
- *Shutdown*: anuncio emitido por un router para indicar que abandonará la red o desactivará determinada pila.

5.1.1 Formato de los Paquetes

- **Cabecera Común:** es el encabezado que viaja al principio de cada PDU del protocolo. Es de tamaño fijo de 16 bytes sin contar los TLVs (ver Figura 5.1). Sus campos son descritos en la Tabla 5.1.

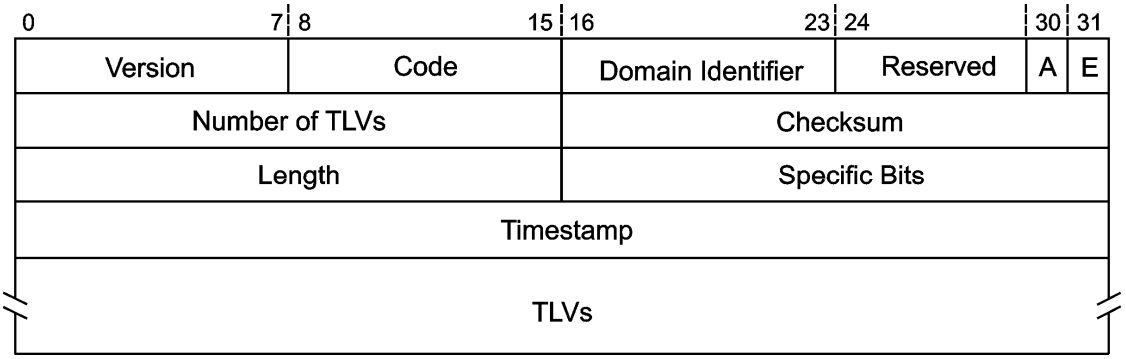

Figura 5.1: Cabecera Común

Campo	Descripción
Version	Versión actual del protocolo, debe ser 1.
Code	Código para identificar el tipo de PDU (ver Tabla 5.2).
Domain Identifier	Identificador del dominio de enrutamiento. Define la pertenencia al mismo dominio de enrutamiento.
Reserved	Reservado para mejoras futuras, debe ir en cero.
A	Bandera para indicar si la autenticación está habilitada. En caso de estar habilitada el primer TLV será de autenticación para cualquier tipo de PDU.
E	Bandera para indicar si el cifrado está habilitado. Debe estar habilitada la bandera de autenticación. Se cifran los TLVs del PDU usando una de las claves de cifrado previamente configuradas por el administrador en el router.
Number of TLVs	Indica cuantos TLVs están contenidos en el PDU.
Checksum	Suma de verificación complemento a 1. Si la bandera E se encuentra habilitada, primero se cifra y luego se calcula el *Checksum*.
Length	Longitud total del PDU.
Specific Bits	Este es un campo de 16 bits y su uso varía dependiendo del tipo de PDU.
Timestamp	Marca de tiempo para evitar ataques de repetición. Se define como el número de segundos transcurridos desde la medianoche del Coordinated Universal Time (UTC) desde el 1 de Enero de 1970.
TLVs	Espacio donde se llevan los TLVs.

Tabla 5.1: Campos de Cabecera Común

Code	PDU
0	Update
1	Request
2	Shutdown

Tabla 5.2: Códigos de PDU

- **Tipos de TLV utilizados en el protocolo:** en la Tabla 5.3 se listan los rangos definidos para cada tipo de TLV. Cada rango tiene un nombre asociado de acuerdo a la funcionalidad que juega, quedando libres algunas posiciones del rango para extensiones futuras.

Type	Nombre
00 – 09	Prefixes
10 – 19	Next Hop Address
20 – 29	Supported Protocols
30 – 39	Authentication
40 – 49	Encryption
50 – 59	Common Services
60 – 255	Reserved

Tabla 5.3: Tipos de TLVs

- **Cabecera para un *Update:*** un *Update* (ver Figura 5.2) es enviado en *unicast* al router que lo solicita como respuesta a un *Request* que tenga el bit I (*Inmediate*) en 1. Este *Update* contiene toda la información de enrutamiento y servicios comunes. También es enviado en *multicast* y de forma periódica según lo indique el parámetro *Update Interval* (30 segundos por defecto). Cuando un router no recibe anuncios de un prefijo por más de *Dead Interval* (90 segundos por defecto), este prefijo debe ser eliminado de la Tabla de Enrutamiento.

  En el caso de que un router tenga que enviar mucha información a sus vecinos a través de mensajes *Update*, puede dividir esta información y enviar varios mensajes sucesivos. La bandera F (*Finish*) permite especificar el último *Update* de una serie. Para eso, el emisor asigna esta bandera en 0 en los primeros mensajes *Update* y en 1 en el último mensaje, para así marcar el final de la actualización. El tamaño del PDU *Update* estará acotado por el MTU asociado a la interfaz por donde se envíe.

  En caso de presentarse un cambio topológico, un router podrá emitir en *multicast* un *Triggered Update* con el bit T (*Triggered Update*) en 1, con el propósito de sólo anunciar dichos cambios antes del próximo *Update* regular. El router que lo recibe debe actualizar su Tabla de Enrutamiento y emitir un *Triggered Update* por sus otras interfaces si existen cambios. Las banderas I o D de cada TLV de prefijo deben estar habilitadas según sea el caso (*Insert/Delete*), no deben ir ambas con el mismo valor para un mismo TLV de prefijo.

  El anuncio de Servicios Comunes en un *Update* se hace periódicamente cada *Common Services Update Interval* (600 segundos por defecto), incluyendo los TLVs *IPv4 DNS* e *IPv6 DNS.* Cuando un router no recibe anuncios de un servidor DNS por más de *Common Services Dead Interval* (1800 segundos por defecto), este servidor DNS debe ser eliminado de la Tabla de Servicios Comunes.

  Posibles TLVs: *MD5 Authentication, MD5 Authentication* con *Encryption, SHA1 Authentication, SHA1 Authentication* con *Encryption, IPv4 Next Hop Address, IPv4 Prefix, IPv6 Next Hop Address, IPv6 Prefix, IPv4 DNS* e *IPv6 DNS.* En la Figura 5.3 se puede observar un dragrama con el orden que estos TLV deben tener en un *Update.*

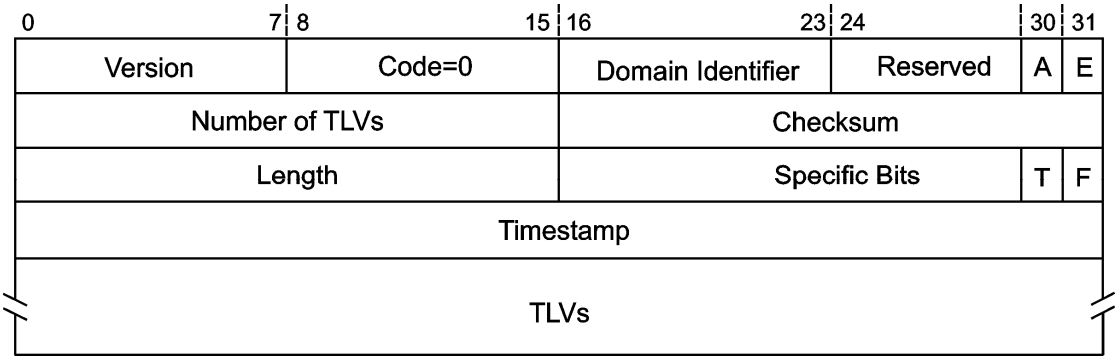

Figura 5.2: Cabecera para un Update

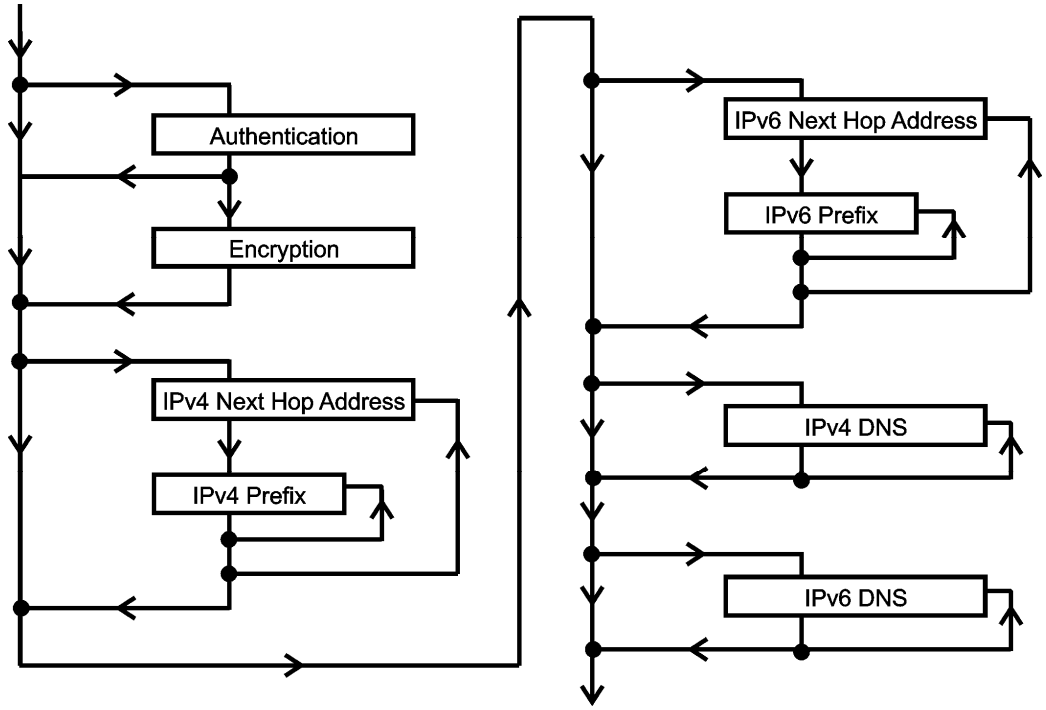

Figura 5.3: Orden de los TLVs en un PDU Update

TLV de *IPv4 Next Hop Address*: indica que los prefijos que se encuentran en los TLVs de *IPv4 Prefix* que siguen, pueden ser alcanzados a través de la dirección especificada en el campo *IPv4 Address* (ver Figura 5.4). Los campos de este TLV son descritos en la Tabla 5.4.

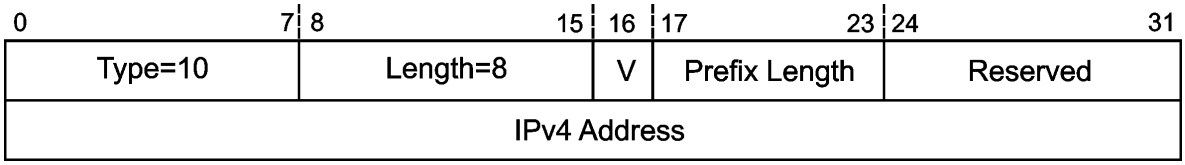

Figura 5.4: TLV de IPv4 Next Hop Address

Campo	Descripción
Type	Debe ser 10.
Length	Longitud total del TLV (8 bytes).
V	Bandera para indicar la verificación si un *Next Hop* pertenece a la misma red. Si vale 1 debe ser verificado que la dirección *IPv4 Address* pertenezca a una red configurada por la interfaz de recepción y que el *Prefix Length* sea el mismo. Cuando vale 0 no debe ser verificado.
Prefix Length	Longitud del Prefijo.
Reserved	Reservado para mejoras futuras, debe ir en cero.
IPv4 Address	Dirección IPv4 del *Next Hop*.

Tabla 5.4: Campos para un TLV de IPv4 Next Hop Address

- **TLV de *IPv4 Prefix:*** contiene la información de un prefijo IPv4 como se muestra en la Figura 5.5. Los campos de este TLV son descritos en la Tabla 5.5.

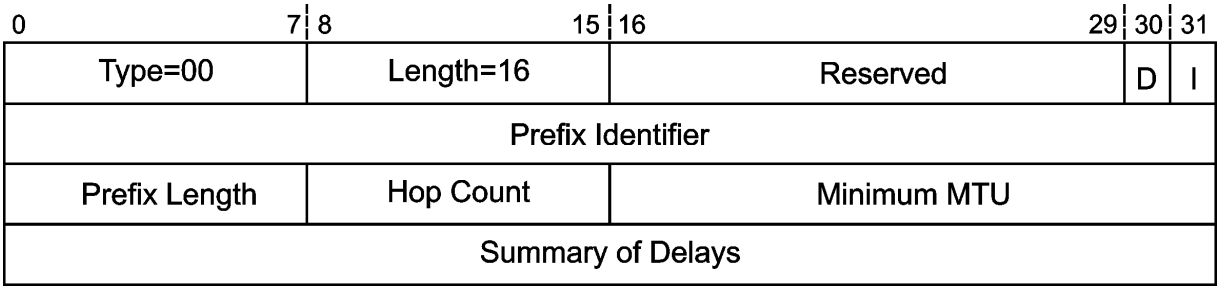

Figura 5.5: TLV de IPv4 Prefix

Campo	Descripción
Type	Debe ser 00.
Length	Longitud total del TLV (16 bytes).
Reserved	Reservado para mejoras futuras, debe ir en cero.
D	Bandera para indicar si el prefijo que se anuncia debe ser borrado (*Delete*). En caso de estar en 1, debe llevar los siguientes valores: *Hop Count* = 0xFF, *Minimun MTU* = 0, *Sum of Delays* = 0xFFFFFFFF.
I	Bandera para indicar si el prefijo que se anuncia debe ser insertado (*Insert*).
Prefix Identifier	Prefijo IPv4.
Prefix Length	Longitud del prefijo.
Hop Count	Número de saltos para alcanzar el prefijo.
Minimum MTU	Mínimo de los MTUs a lo largo de la ruta para alcanzar el prefijo.
Sum of Delays	Sumatoria de los retardos de la ruta.

Tabla 5.5: Campos para un TLV de IPv4 Prefix

- **TLV de *IPv4 DNS*:** indica una lista de servidores DNS correspondientes a una red IPv4 (ver Figura 5.6). Los campos de este TLV son descritos en la Tabla 5.6.

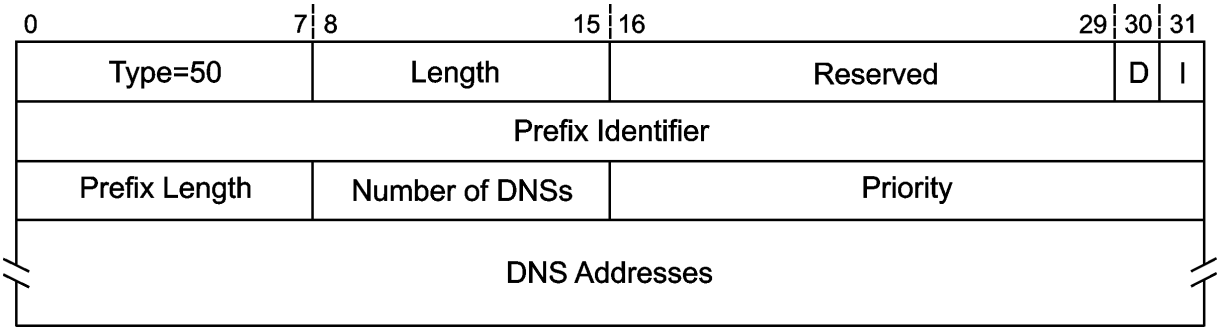

Figura 5.6: TLV de IPv4 DNS

Campo	Descripción
Type	Debe ser 50.
Length	Longitud total del TLV (variable).
Reserved	Reservado para mejoras futuras, debe ir en cero.
D	Bandera para indicar que la lista de servidores DNSs que se anuncia debe ser borrada (*Delete*).
I	Bandera para indicar que la lista de servidores DNSs que se anuncia debe ser insertada (*Insert*).
Prefix Identifier	Prefijo IPv4 que deben utilizar los servidores DNSs anunciados.
Prefix Length	Longitud del prefijo.
Number of DNSs	Indica cuantos DNSs están contenidos en el TLV.
Priority	Establece la prioridad de los DNSs anunciados. Utilizado para levantar la indeterminación creada por anuncios distintos para una misma red, el valor positivo más pequeño tiene mayor prioridad.
DNS Addresses	Direcciones IPv4 de DNSs.

Tabla 5.6: Campos para un TLV de IPv4 DNS

- **TLV de *IPv6 Next Hop Address*:** indica que los prefijos que se encuentran en los TLVs de *IPv6 Prefix* que siguen, pueden ser alcanzados a través de la dirección especificada en el campo *IPv6 Address* (ver Figura 5.7). Los campos de este TLV son descritos en la Tabla 5.7.

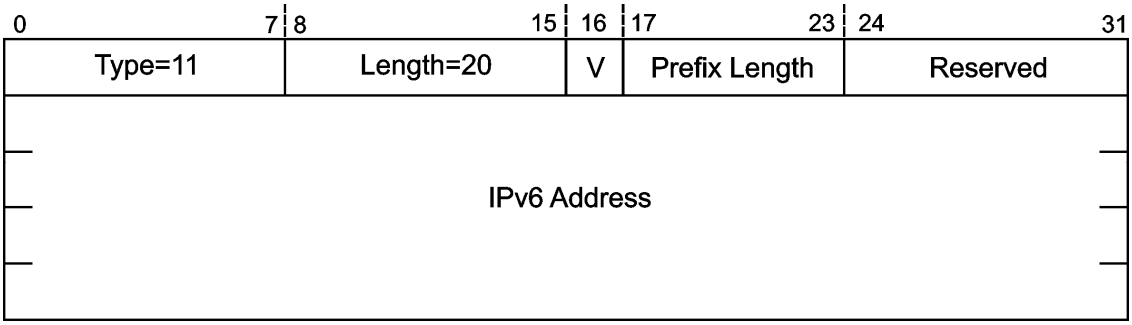

Figura 5.7: TLV de IPv6 Next Hop Address

Campo	Descripción
Type	Debe ser 11.
Length	Longitud total del TLV (20 bytes).
V	Bandera para indicar la verificación si un *Next Hop* pertenece a la misma red. Si vale 1 debe ser verificado que la dirección *IPv6 Address* pertenezca a una red configurada por la interfaz de recepción y que el *Prefix Length* sea el mismo. Cuando vale 0 no debe ser verificado.
Prefix Length	Longitud del Prefijo.
Reserved	Reservado para mejoras futuras, debe ir en cero.
IPv4 Address	Dirección IPv6 del *Next Hop*.

Tabla 5.7: Campos para un TLV de IPv6 Next Hop Address

- **TLV de *IPv6 Prefix*:** contiene la información de un prefijo IPv6 como se muestra en la Figura 5.8. Los campos de este TLV son descritos en la Tabla 5.8.

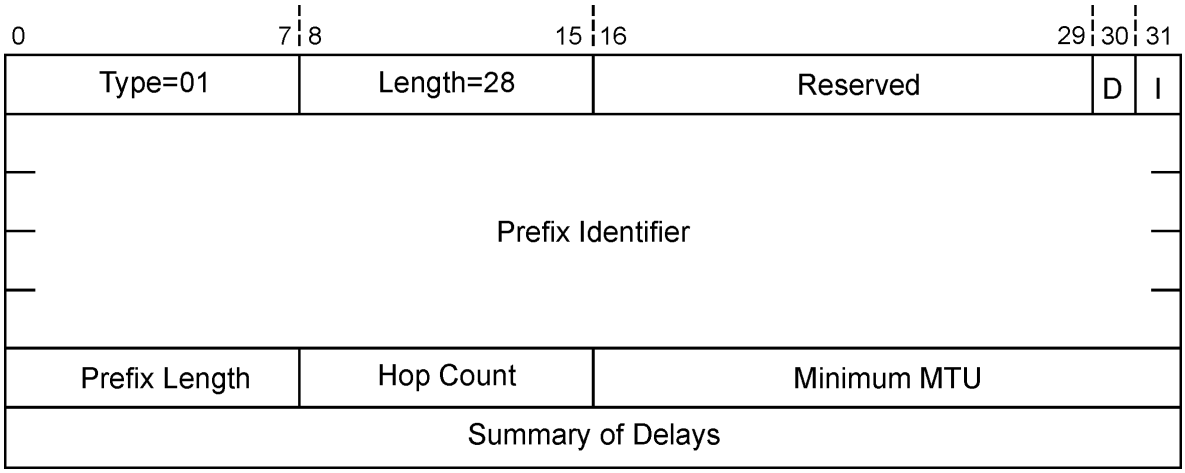

Figura 5.8: TLV de IPv6 Prefix

Campo	Descripción
Type	Debe ser 01.
Length	Longitud total del TLV (28 bytes).
Reserved	Reservado para mejoras futuras, debe ir en cero.
D	Bandera para indicar si el prefijo que se anuncia debe ser borrado (*Delete*). En caso de estar en 1, debe llevar los siguientes valores: *Hop Count* = 0xFF, *Minimun MTU* = 0, *Sum of Delays* = 0xFFFFFFFF.
I	Bandera para indicar si el prefijo que se anuncia debe ser insertado (*Insert*).
Prefix Identifier	Prefijo IPv6.
Prefix Length	Longitud del prefijo.
Hop Count	Número de saltos para alcanzar el prefijo.
Minimum MTU	Mínimo de los MTUs a lo largo de la ruta para alcanzar el prefijo.
Sum of Delays	Sumatoria de los retardos de la ruta.

Tabla 5.8: Campos para un TLV de IPv6 Prefix

- **TLV de *IPv6 DNS*:** indica una lista de servidores DNS correspondientes a una red IPv6 (ver Figura 5.9). Los campos de este TLV son descritos en la Tabla 5.9.

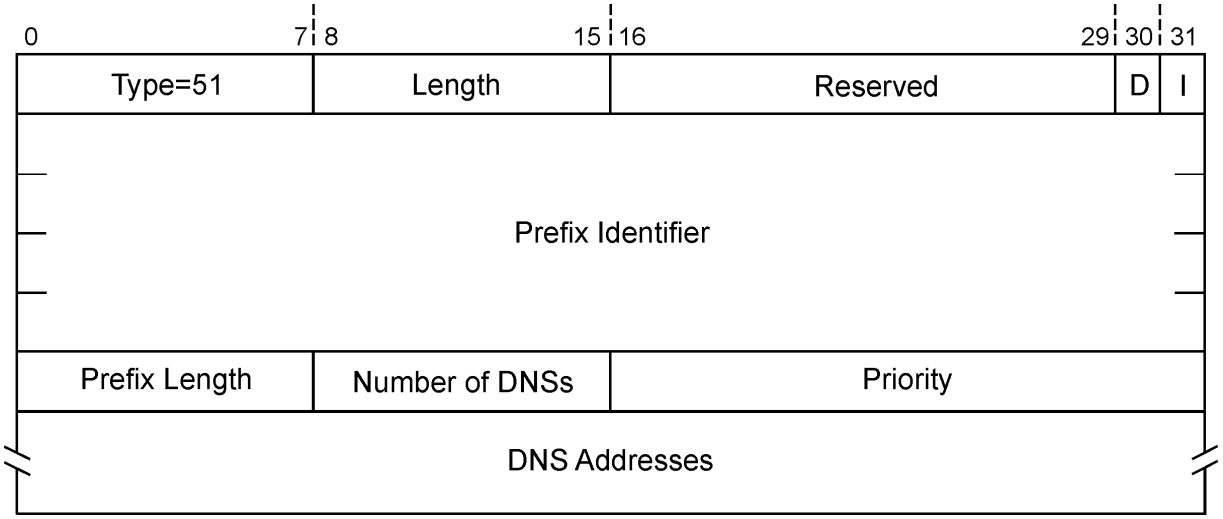

Figura 5.9: TLV de IPv6 DNS

Campo	Descripción
Type	Debe ser 51.
Length	Longitud total del TLV (variable).
Reserved	Reservado para mejoras futuras, debe ir en cero.
D	Bandera para indicar que la lista de servidores DNSs que se anuncia debe ser borrada (*Delete*).
I	Bandera para indicar que la lista de servidores DNSs que se anuncia debe ser insertada (*Insert*).
Prefix Identifier	Prefijo IPv6 que deben utilizar los servidores DNSs anunciados.
Prefix Length	Longitud del prefijo.
Number of DNSs	Indica cuantos DNSs están contenidos en el TLV.
Priority	Establece la prioridad de los DNSs anunciados. Utilizado para levantar la indeterminación creada por anuncios distintos para una misma red, el valor positivo más pequeño tiene mayor prioridad.
DNS Addresses	Direcciones IPv6 de DNSs.

Tabla 5.9: Campos para un TLV de IPv6 DNS

- **Cabecera para un *Request*:** un *Request* es enviado por todas las interfaces de un router cuando es encendido, cuando se activa la pila IPv4 o IPv6, o por una interfaz que se acaba de activar. Se debe enviar tres veces y es usado para solicitar a los routers adyacentes la información de enrutamiento de los protocolos especificados en el TLV de *Supported Protocols* (ver Figura 5.10).

  Los *Request* iniciales deben llevar el bit I (*Inmediate*) en 1 para indicar que la solicitud debe ser respondida de inmediato. Posteriormente serán enviados una vez cada *Request Interval* (60 segundos por defecto) con el bit I en 0 para

especificar que no deben ser respondidos de inmediato sino en el próximo *Update* periódico. Esto último permite que dos o más routers que pertenezcan a un mismo segmento de red conozcan los protocolos que soportan en dicho segmento.

Posibles TLVs: *MD5 Authentication, SHA1 Authentication, Supported Protocols*. En la Figura 5.11 se puede observar un dragrama con el orden que estos TLV deben tener en un *Update.*

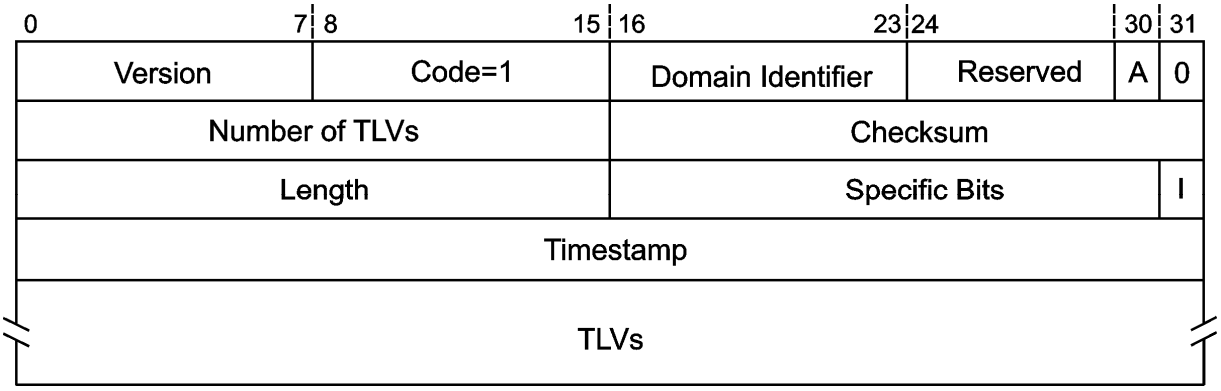

Figura 5.10: Cabecera para un Request

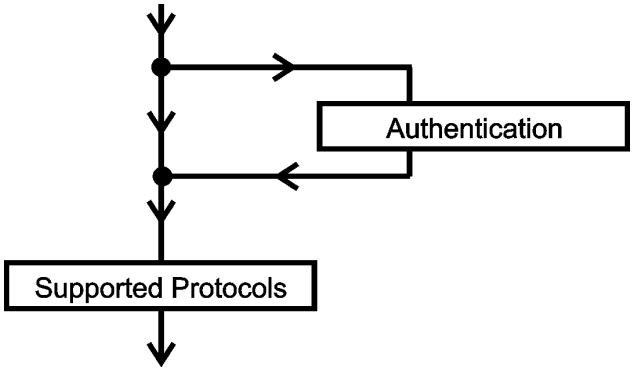

Figura 5.11: Orden de los TLVs en un PDU Request

- **TLV de *Supported Protocols*:** usado para informar sobre los protocolos de red soportados a los routers adyacentes mediante un *Request* o periódicamente con los *Update* (ver Figura 5.12). Los campos de este TLV son descritos en la Tabla 5.10.

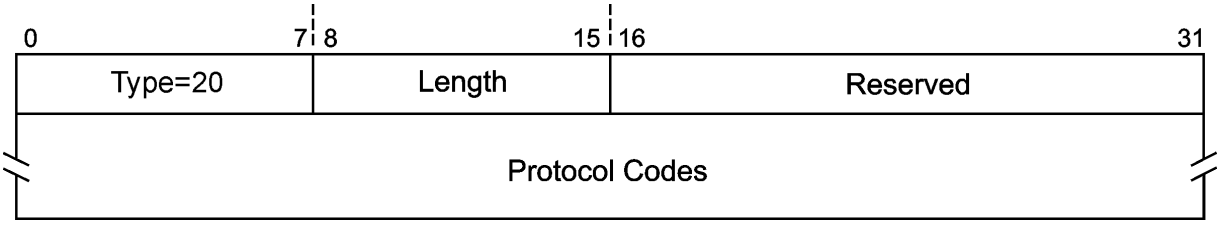

Figura 5.12: TLV de Supported Protocols

Campo	Descripción
Type	Debe ser 20.
Length	Longitud total del TLV (variable). Debe ser múltiplo de 4.
Reserved	Reservado para mejoras futuras, debe ir en cero.
Protocol Codes	Códigos de protocolos soportados (ver Tabla 5.11). De ser necesario se debe rellenar con ceros para ser múltiplo de 4 bytes.

Tabla 5.10: Campos para un TLV de Supported Protocols

Protocolo	Código
IPv4	0xCC
IPv6	0x8E
Padding	0x00

Tabla 5.11: Códigos de Protocolos

- **Cabecera para un *Shutdown*:** mensaje enviado para indicar a los routers adyacentes que el router emisor no permanecerá más en el segmento, acelerando el tiempo de convergencia. Este mensaje debe ser enviado tres veces con 1 segundo de separación. Es posible anunciar la desactivación de la pila IPv4 o IPv6 (ver Figura 5.13).

  Posibles TLVs: *MD5 Authentication, SHA1 Authentication*, *IPv4 Next Hop Address, IPv6 Next Hop Address.* En la Figura 5.14 se puede observar un dragrama con el orden que estos TLV deben tener en un *Shutdown.*

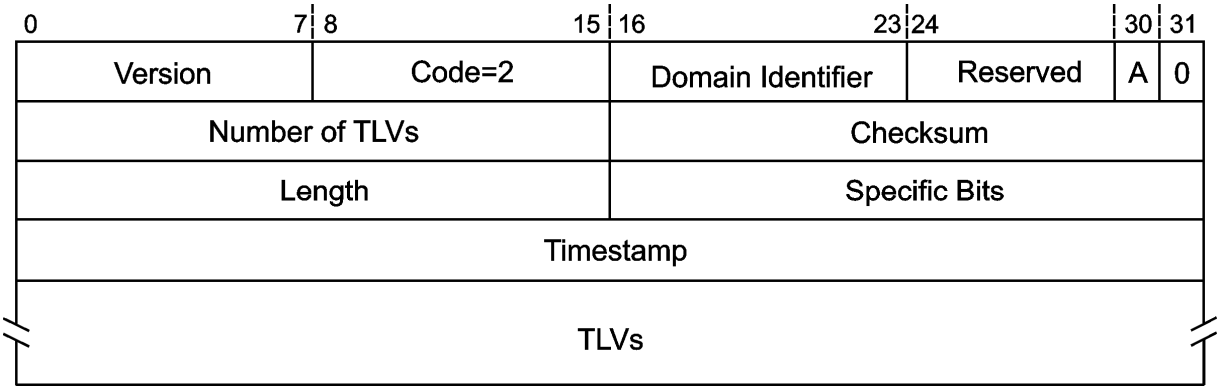

Figura 5.13: Cabecera para un Shutdown

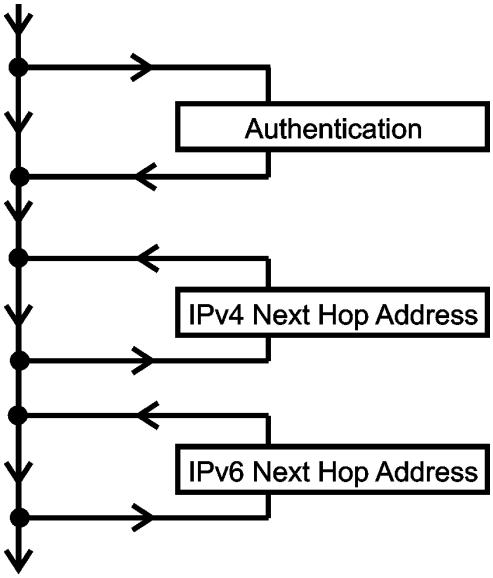

Figura 5.14: Orden de los TLVs en un PDU Shutdown

- **TLV de *Clear Text Authentication*:** puede ser usado para establecer comunicación autenticada entre dos routers (ver Figura 5.15). Los campos de este TLV son descritos en la Tabla 5.12.

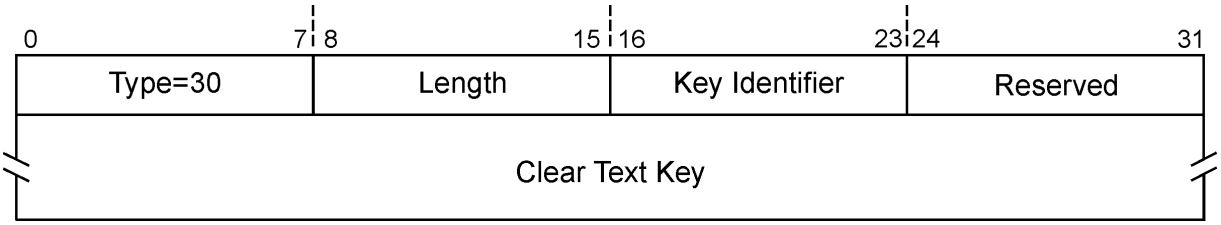

Figura 5.15: TLV de Clear Text Authentication

Campo	Descripción
Type	Debe ser 30.
Length	Longitud total del TLV (variable). Debe ser múltiplo de 4.
Key Identifier	Identificador de clave de autenticación. Se puede establecer una lista de claves y este campo indica cuál de ellas se está usando. La primera clave está identificada por 0.
Reserved	Reservado para mejoras futuras, debe ir en cero.
Clear Text Key	Clave en claro. Se recomienda usar este tipo de autenticación con fines de depuración. De ser necesario, se debe rellenar con ceros para ser múltiplo de 4 bytes.

Tabla 5.12: Campos para un TLV de Clear Text Authentication

- **TLV de *MD5 Authentication*:** contiene información referente a los datos de autenticación cuando se utiliza el algoritmo MD5 (ver Figura 5.16). Los campos de este TLV son descritos en la Tabla 5.13.

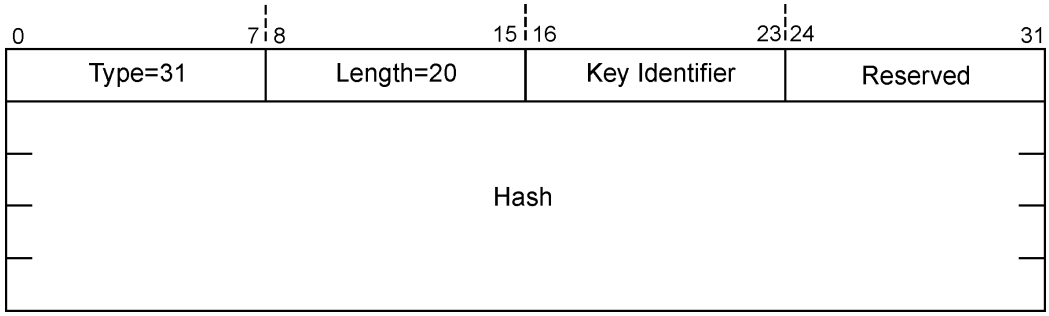

Figura 5.16: TLV de MD5 Authentication

Campo	Descripción
Type	Debe ser 31.
Length	Longitud total del TLV (20 bytes).
Key Identifier	Identificador de clave de autenticación. Se puede establecer una lista de claves de autenticación para MD5 y este campo indica cuál de ellas se está usando. La primera clave está identificada por 0.
Reserved	Reservado para mejoras futuras, debe ir en cero.
Hash	Es el resultado de aplicar el algoritmo MD5 al PDU concatenado con la clave de autenticación. Luego el resultado se almacena en el campo Hash del TLV de autenticación (que debe encontrarse de primero), asegurando la integridad y autenticidad del mensaje. Para efectos del cálculo, el campo Hash del TLV de *MD5 Authentication* es inicializado en cero.

Tabla 5.13: Campos para un TLV de MD5 Authentication

- **TLV de *SHA1 Authentication*:** contiene información referente a los datos de autenticación cuando se utiliza el algoritmo SHA1 (ver Figura 5.17). Los campos de este TLV son descritos en la Tabla 5.14.

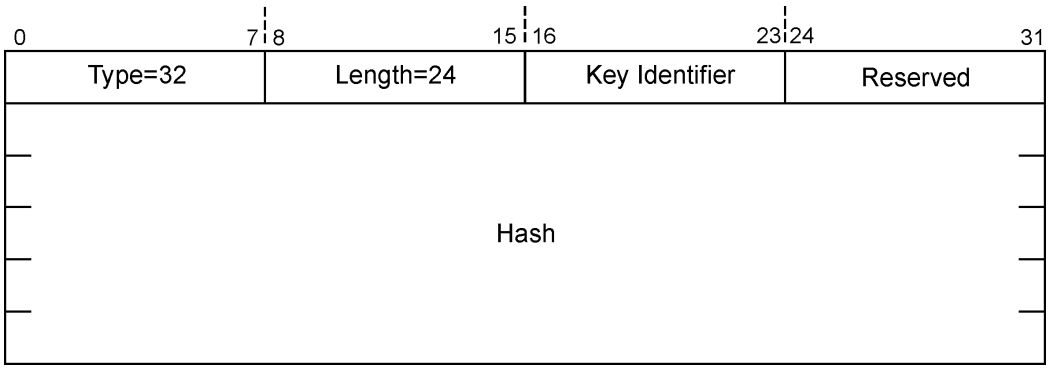

Figura 5.17: TLV de SHA1 Authentication

Campo	Descripción
Type	Debe ser 32.
Length	Longitud total del TLV (24 bytes).
Key Identifier	Identificador de clave de autenticación. Se puede establecer una lista de claves de autenticación para SHA1 y este campo indica cuál de ellas se está usando. La primera clave está identificada por 0.
Reserved	Reservado para mejoras futuras, debe ir en cero.
Hash	Es el resultado de aplicar el algoritmo SHA1 al PDU concatenado con la clave de autenticación. Luego el resultado se almacena en el campo Hash del TLV de autenticación (que debe encontrarse de primero), asegurando la integridad y autenticidad del mensaje. Para efectos del cálculo, el campo Hash del TLV de *SHA1 Authentication* es inicializado en cero.

Tabla 5.14: Campos para un TLV de SHA1 Authentication

- **TLV de *Encryption*:** en este TLV viaja la información concerniente al algoritmo usado para cifrar el PDU (ver Figura 5.18). Los campos de este TLV son descritos en la Tabla 5.15.

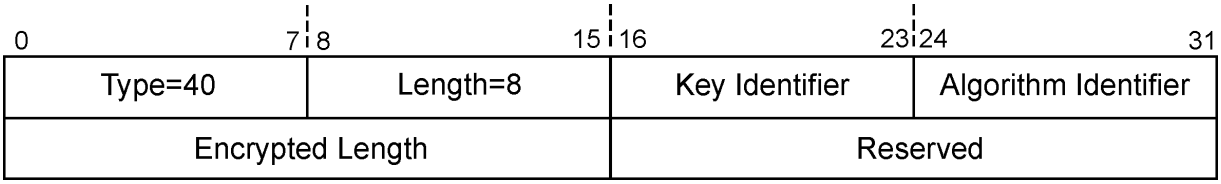

Figura 5.18: TLV de Encryption

Campo	Descripción
Type	Debe ser 40.
Length	Longitud total del TLV (8 bytes).
Key Identifier	Identificador de clave de cifrado. Se puede establecer una lista de claves de cifrado para cada algoritmo y este campo indica cuál de ellas se está usando. La primera clave está identificada por 0.
Algorithm Identifier	Identificador de algoritmo de cifrado. En la versión actual (versión 1), el protocolo soporta AES, DES y 3DES (ver Tabla 5.16).
Encrypted Length	Longitud de los datos cifrados

Tabla 5.15: Campos para un TLV de Encryption

Algorithm Identifier	Algoritmo
00	AES
01	DES
02	3DES

Tabla 5.16: Identificadores de Algoritmos

5.1.2 Extensibilidad del Protocolo

Siguiendo el diseño estructural que utilizan EIGRP e IS-IS, con respecto al uso de TLVs en el formato de los mensajes, DBRP también hace uso de los mismos con el objetivo de poder extender las características y funcionalidades del protocolo sin que sean necesarias modificaciones drásticas en el formato de los mensajes.

En la Figura 5.19 se puede observar este esquema, donde la cabecera común (*Common Header*) es de tamaño fijo y contempla un campo que especifica el número TLVs del mensaje. Cada TLV puede variar en su longitud y puede contener determinada información dependiendo de su tipo. De este modo, se pueden incluir tantos TLVs en un mensaje como permita el MTU (*Maximum Transmission Unit*) del enlace.

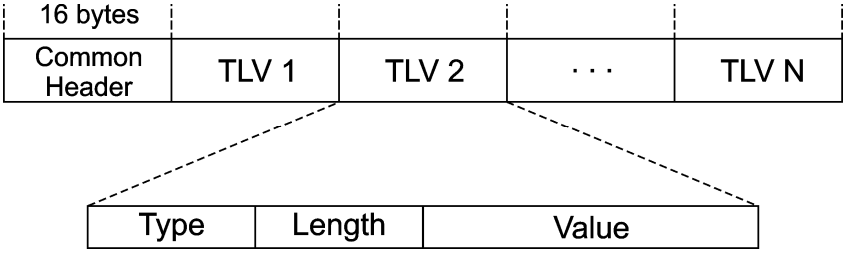

Figura 5.19: Extensión Mediante el uso de TLVs

En la Figura 5.20, Figura 5.21 y Figura 5.22 se puede observar la estructura de los PDUs *Update*, *Request* y *Shutdown* con sus TLVs, respectivamente.

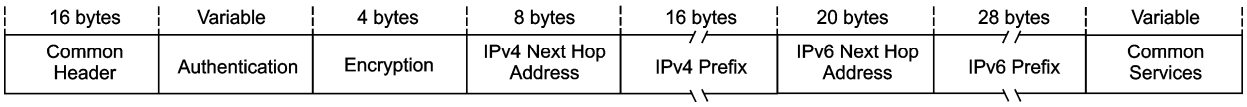

Figura 5.20: PDU Update

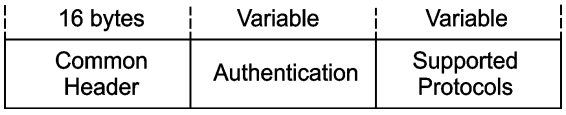

Figura 5.21: PDU Request

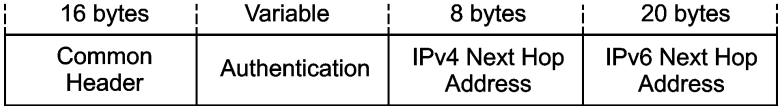

Figura 5.22: PDU Shutdown

5.2 Servicios Comunes

Con el objetivo de simplificar las tareas de los administradores, DBRP es capaz de propagar información de servicios comunes tal como servidores DNSs, servidores de impresión, servidores SIP, servidores NTP y otros servicios que se deseen configurar de forma automática. En esta versión de DBRP (v1), sólo se implementa la propagación de servidores DNSs, quedando a disposición de otros desarrolladores la inclusión de nuevos servicios.

La información de los DNSs es anunciada en mensajes *Update*. De este modo, un administrador que desee configurar de manera automática los hosts pertenecientes a una red, puede incluir la información de los servidores DNSs y su prioridad, en mensajes *Update*. Un router que recibe un anuncio de servicios comunes debe verificar si está dirigido a él, de ser asi, el router almacena la información recibida en su Tabla de Servicios Comunes. En caso contrario debe reenviarla a la red que corresponde según indica el TLV de *IPv4 DNS* o *IPv6 DNS*.

5.3 Soporte de IPv4 e IPv6

A lo largo de la evolución de los protocolos de enrutamiento tanto de vector de distancia como de estado de enlace, ha sido necesario hacer modificaciones o extensiones a los protocolos existentes para dar soporte a nuevas tecnologías tal como IPv6. En el caso de RIP, fue necesario desarrollar una nueva versión llamada RIPng que únicamente soporta IPv6. En el caso de OSPF sucede lo mismo, tuvo que ser implementado OSPFv3 para dar soporte a IPv6, ya que OSPFv2 no lo hacía. Cisco Systems diseñó inteligentemente EIGRP a base de TLVs y lo pudo extender con TLVs adicionales para dar soporte a IPv6. La nueva versión del protocolo se llama EIGRPv6 y conserva el soporte de IPv4 sin necesidad de rehacer un nuevo protocolo.

En el diseño de DBRP se hace uso inteligente de los TLVs, dando soporte tanto a IPv4 como a IPv6. La extensión posterior a otros protocolos de red es facilitada por la inclusión de nuevos TLVs y al hecho de no depender de la capa de red para operar. En la Figura 5.23 se puede observar que DBRP viaja encapsulado en la capa de enlace. Con esto se logra la independencia de la capa de red, siendo posible en futuras versiones incluir otros protocolos de capa de red.

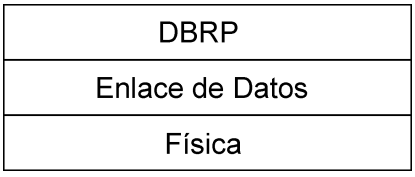

Figura 5.23: Ubicación de DBRP en la Pila de Protocolos

5.4 Elección de la Métrica

RIP utiliza la cantidad de saltos como métrica. IGRP y EIGRP se enfocan en características del enlace, tales como el ancho de banda y el retardo, donde ambos factores son combinados en una fórmula y expresados como un único valor.

Si se toma en cuenta la suma de los retardos asociados a los enlaces por los cuales debe transitar un mensaje, esto representa una medida más justa con respecto a que si se compara únicamente con el número de nodos que el paquete debe cruzar; es decir, si un prefijo está a 3 saltos de distancia y los enlaces son muy lentos, es preferible seleccionar una ruta de 5 saltos cuyos enlaces sean más rápidos. Es por esto que DBRP toma en cuenta el retardo como única métrica.

El administrador puede asignar un retardo al router en base a la congestión que pueda tener y que no depende del retardo de los enlaces. Puede tener un rango de 0 - 16.777.215 ns, descrito en la Subsección 7.2.1. El retardo total para un prefijo viene dado por la siguiente expresión:

Retardo (Total) = Retardo (Enlace) + Retardo (Router)

A su vez, el retardo del enlace se compone de serialización y propagación: (1) el tiempo requerido para insertar datos en el medio es conocido como tiempo de serialización. Este tiempo se determina en base al volumen de datos y velocidad del enlace y (2) el tiempo requerido para cruzar el medio es conocido como tiempo de propagación y depende de la longitud del enlace.

Por ejemplo, en FastEthernet (100 Mbps), el cálculo sería el siguiente:

Retardo (Enlace) = Retardo (Serialización) + Retardo (Propagación)

Como la trama más larga de FastEthernet es de 8+14+1500+4 = 1526 bytes

- *8 bytes de sincronización*
- *14 de cabecera Ethernet*
- *1500 de MTU*
- *4 de CRC-32*

Entonces el valor más grande del retardo de serialización sería:

Retardo (Serialización) = (1526 * 8) / (100E6) = 0,00012208 seg = 122.080 ns

Si la señal se propaga al 80% de la velocidad de la luz (300.000.000 m/s), y el cable más largo es de 100 metros, entonces el retardo de propagación más grande es de:

Retardo (Propagación) = 100 / (0,8 * 300.000.000) = 417 ns

Por tanto, el retardo total para FastEthernet sería:

Retardo (Enlace) = 122.080 ns + 417 ns = 122.497 ns

La Tabla 5.17 muestra retardos asociados a otros tipos de enlaces.

Link Type	Link Speed	Delay
FastEthernet	100 Mbps	122.080 ns
Ethernet	10 Mbps	1.220.800 ns

Tabla 5.17: Retardo de los Enlaces

5.5 Tiempo de Convergencia

Se dice que los routers de una red han convergido cuando todos poseen la misma imagen del estado de la red. El tiempo que transcurre para que todos los routers hallan convergido se llama tiempo de convergencia. Una característica importante añadida a este protocolo, que contribuye con la disminución del tiempo de convergencia, es el uso de los mensajes *Request* y *Shutdown.* Para complementar los *Update* regulares, los mensajes *Triggered Update* permiten anunciar cambios de inmediato, sin necesidad de esperar por el próximo *Update* síncrono.

5.6 Ciclos de Enrutamiento

Existen diversos mecanismos para evitar ciclos de enrutamiento. DBRP implementa *Split Horizon* señalando que si un prefijo es aprendido por determinada interfaz, ese prefijo no debe anunciarse por esa misma interfaz.

También se hace uso de *Holddown*, ignorando durante dos veces el *Update Interval*, anuncios de un prefijo con peor métrica provenientes del *Next Hop*.

5.7 Tamaño del Sistema Autónomo

En protocolos como RIP, el tamaño del dominio de enrutamiento es un problema, ya que al existir un prefijo a más de 15 saltos se considera inalcanzable. Este no es el caso de DBRP, en donde se pueden alcanzar destinos de hasta 100 saltos de distancia como lo permite el valor por defecto de *Hop Count*. Este valor se considera lo suficientemente grande para un protocolo IGP.

5.8 Seguridad

Cuando un router anuncia a otro router determinada información de enrutamiento, dicha información podría ser vulnerable a muchos niveles, ya sea por ataques o por alteración de los datos por problemas físicos en el enlace. Empleando mecanismos como la autenticación y el cifrado es posible contrarrestar estas situaciones [10].

5.8.1 Autenticación

La autenticación establece una relación de confianza entre dos o más entes, de manera que la información que se intercambie entre ellos no pueda ser suplantada por entes no autorizados. DBRP permite tres tipos de autenticación:

- *Clear Text Authentication*: permite establecer confianza entre routers sin la utilización de ningún algoritmo de *Hashing*, compartiendo una misma lista de claves previamente configuradas por el administrador en cada router. Cada clave debe identificarse con un *Key Identifier*. No se recomienda este método, ya que la clave viaja en claro. Su uso se debería limitar a escenarios de depuración.
- *MD5 Authentication*: permite establecer confianza entre routers mediante la utilización del algoritmo MD5.
- *SHA1 Authentication*: permite establecer confianza entre routers mediante la utilización del algoritmo SHA1.

Pasos para enviar un mensaje autenticado:

- Cuando un router envía un mensaje autenticado a otro router, debe poner en 1 el valor de la bandera *A* (*Authentication*) en la cabecera común del PDU.
- Luego elige aleatoriamente una de las claves de su lista correspondiente al tipo de autenticación configurada.
- En caso de autenticación en claro, se copia la clave seleccionada en el campo *Clear Text Key* del TLV de autenticación, antes de enviar el mensaje.
- En caso de autenticación MD5 y SHA1, se calcula el *Hash* del TLV de autenticación concatenando el PDU entero con la clave que indique el *Key Identifier*, generando así el resultado que se copia en el campo *Hash* del TLV antes de enviar el mensaje.

Pasos para recibir un mensaje autenticado:

- Cuando el router destino recibe el mensaje, verifica el valor de la bandera *A* en la cabecera común para saber si es un mensaje autenticado.
- En caso de estar autenticado, determina el tipo de autenticación según el campo *Type* del primer TLV de autenticación.
- Si el TLV es de *Clear Text Authentication*, compara el valor del campo *Clear Text Key* con la clave asociada a la posición del *Key Identifier* de la lista de claves

para este tipo de autenticación; estos valores deben coincidir para que el mensaje sea tomado en cuenta.

- Si el TLV es de *MD5 Authentication* o *SHA1 Authentication*, se debe calcular nuevamente un hash, concatenando la clave correspondiente según lo indique el *Key Identifier* con el PDU recibido, y poniendo el campo *Hash* de dicho PDU en 0. Luego se compara el hash recién calculado con el hash del PDU recibido, estos valores deben coincidir para que el mensaje se considere válido.

5.8.2 Cifrado

La criptografía es empleada para hacer cifrar las comunicaciones privadas a entes remotos. Los datos privados pueden hacerse ininteligibles a entes no autorizados a través del cifrado, el cual utiliza complejos algoritmos que convierten el mensaje original o plano, en un mensaje codificado llamado texto cifrado. DBRP permite llevar a cabo una comunicación cifrada a través de algoritmos como AES (*Advanced Encryption Standard*), DES (*Data Encryption Standard*) y 3DES (*Triple DES*). Este mecanismo de seguridad sólo puede ser usado cuando está presente la autenticación.

Con el objetivo de no invertir tiempo de procesamiento descifrando un mensaje que posiblemente no es válido, el PDU es autenticado posterior a su cifrado, de manera que cuando el mensaje sea recibido en el destino se pueda verificar su autenticidad antes de ser descifrado.

El procedimiento para cifrar un mensaje es el siguiente:

- Se construye el PDU asignando en 1 el valor de la bandera *E* (*Encryption*) de la cabecera común, y estableciendo el TLV de autenticación con hash en valor 0.
- Se añade el TLV de *Encryption* eligiendo aleatoriamente la clave, ello a través del campo *Key Identifier.*
- Se aplica el cifrado a los TLVs subsiguientes al TLV de *Encryption* y se genera el texto cifrado de los mismos, almacenando en el campo *Encrypted Length* la longitud resultante del contenido cifrado.
- Ahora se tiene un PDU conformado por la Cabecera Común, los TLVs de autenticación y cifrado, seguido del texto cifrado generados en el paso anterior, y a todo esto se le aplica el mecanismo de autenticación ya descrito en la Subsección 5.8.1.

El procedimiento para descifrar un mensaje es el siguiente:

- Cuando es recibido el mensaje en el destino, se verifica que las banderas A y E estén ambas en 1, de lo contrario el mensaje no será tomado en cuenta, ya que no puede existir cifrado sin autenticación.
- Se procede a verificar la autenticidad del mensaje, tal como se describe en la Subsección 5.8.1.
- Si se considera auténtico el mensaje, se procede a descifrar el texto cifrado con el algoritmo y clave indicados en el TLV de *Encryption*.

6. Marco Metodológico

Para alcanzar los objetivos planteados en el Capítulo 2, es necesario definir un esquema o metodología de trabajo que permita el desarrollo coherente 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 del protocolo de enrutamiento y la herramienta de configuración.

6.1 Adaptación de la Metodología de Desarrollo

La programación extrema o XP [1] (*eXtreme Programming*) es un método de Programación Ágil [3] que se diferencia de las metodologías tradicionales principalmente porque pone más énfasis en la adaptabilidad que en la previsibilidad. XP considera que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos.

Se puede considerar la programación extrema como la adopción de las mejores metodologías de desarrollo de acuerdo a lo que se pretende llevar a cabo con el proyecto, y aplicarlo de manera dinámica durante el ciclo de vida del software.

Los valores de programación extrema que principalmente se explotan son el desarrollo iterativo e incremental, detección y corrección constante de errores antes de cada entrega a través de pruebas modulares y continuas.

Esta metodología hace énfasis en las siguientes cuatro fases: Planificación, Diseño, Codificación y Pruebas[3].

6.1.1 Planificación

En esta primera fase son generados los requerimientos, se planifica un tiempo estimado para cada uno de ellos, luego serán desarrollados mediante iteraciones que cubren pequeñas funcionalidades o características requeridas. La combinación de estas iteraciones provee un producto final funcional.

Parte de esta planificación consiste en organizar los requerimientos en función de su prioridad, de esta manera se tiene una visión general de las actividades que se pueden hacer de forma lineal o en paralelo.

[3] http://www.extremeprogramming.org/rules.html

6.1.2 Diseño

Una iteración en programación XP comienza con el diseño y los principales lineamientos de esta etapa son:

- Aplicar simplicidad al expresar las cosas solo una vez, sin añadir funcionalidades anticipadamente.
- Utilizar un estándar en la nomenclatura de nombres de variables y métodos que mantenga la coherencia entre los miembros del equipo de desarrollo durante el trabajo.
- Organizar las ideas en estructuras lógicas de programación.
- Crear soluciones puntuales o programas sencillos para un problema en específico que haya sido planteado dentro de los requerimientos.

6.1.3 Codificación

La programación constituye la fase con mayor prioridad por sobre las demás actividades, ya que el objetivo es obtener resultados sustanciales al final de cada jornada. Los estándares relacionados a esta fase incluyen:

- Desarrollo del código basado en metáforas y estándares previamente acordados.
- Programación en pareja utilizando una estación de programación con miras a producir código de alta calidad en igual o menor costo.
- Organización de un horario de trabajo semanal de unas 40 horas y cumplirlo a cabalidad, asegurando el trabajo óptimo pero sin sobrepasar las facultades mentales y físicas del equipo de desarrollo.
- Integración frecuente del código con los módulos ya probados y funcionales.

6.1.4 Pruebas

A través de pruebas unitarias se busca corregir cada nuevo desarrollo, de esta manera eliminar errores puntuales antes de integrar con otros módulos. Una vez solucionados los problemas de programación se hacen pruebas con el usuario basadas en sus requerimientos, con el fin de validar el código.

6.2 Tecnologías a Utilizar

Será utilizado (1) el lenguaje de programación C++, por su robustez, amplia documentación y soporte nativo para las bibliotecas Qt y Socket, (2) las plataformas de virtualización VirtualBox y VMware para emular routers, (3) GDB para depuración del código, (4) Libssl-dev para proveer funciones criptográficas, (5) Qt y Qt-Designer para la creación de la interfaz gráfica, y (6) Wireshark y Tcpdump para el análisis de los mensajes intercambiados entre los routers.

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

- C++[4]: Lenguaje de programación híbrido que soporta la programación estructurada y orientado a objetos. Uno de los compiladores libres de C++ es el de GNU, el compilador g++.
- GDB[5] (*GNU Debugger*): Depurador estándar para el sistema operativo GNU/Linux. Es un depurador portable que se puede utilizar en varias plataformas (Unix y Windows) y funciona para varios lenguajes de programación, basado en licencia GPL.
- Libssl-dev: Robusto paquete orientado a ambientes Unix que provee herramientas de administración y bibliotecas de desarrollo relacionadas con la criptografía, de código abierto basado en licencia Apache.
- Qt[6]: Biblioteca multiplataforma ampliamente usada para desarrollar aplicaciones con una interfaz gráfica de usuario. Sin embargo también permite el desarrollo de programas sin interfaz gráfica como herramientas para la línea de comandos y consolas para servidores. Es una iniciativa desarrollada originalmente por Trolltech.
- Qt-Designer: Herramienta mediante la cual se puede realizar el diseño de aplicaciones basadas en GUI (*Graphical User Interface*) utilizando las bibliotecas de Qt.
- Wireshark[7]: Antes conocido como Ethereal, es un analizador de protocolos utilizado para solucionar problemas en redes de comunicaciones, ampliamente empleado como herramienta didáctica para la educación. Posee una interfaz gráfica y muchas opciones de organización y filtrado de información. Permite ver todo el tráfico que pasa a través de una red Ethernet, aunque es compatible con algunas otras tecnologías (e.g., WiFi, Point-to-Point), estableciendo la configuración de las interfaces de red en modo promiscuo.
- Tcpdump[8]: Herramienta en línea de comandos cuya utilidad principal es analizar el tráfico que circula por la red, muy similar a Wireshark sólo que sin interfaz gráfica.
- VirtualBox[9]: Producto de virtualización orientado a estaciones de trabajo, desarrollado inicialmente por Innotek GmbH, para arquitecturas x86 y AMD64/Intel64, de uso empresarial y doméstico. Proporciona un ambiente de ejecución similar al de un computador físico con CPU, BIOS, tarjeta gráfica, memoria RAM, tarjetas de red, conexiones USB, discos duros, etc. Es robusto y se encuentra disponible gratuitamente bajo licencia GPL versión 2.

[4] http://www2.research.att.com/~bs/C++.html
[5] http://www.gnu.org/s/gdb
[6] http://qt.nokia.com
[7] http://www.wireshark.org
[8] http://www.tcpdump.org
[9] http://www.virtualbox.org

- VMware[10] (workstation): Sistema de virtualización por software, que proporciona las mismas ventajas que VirtualBox sólo que bajo licencia comercial.

6.3 Prototipo General de Interfaz

Con el objetivo de proporcionar un entorno visual sencillo para permitir la comunicación entre el administrador y el protocolo, se desarrolló una interfaz gráfica utilizando Qt-Designer y las bibliotecas de Qt, que permite configurar los parámetros generales del protocolo, ver la Tabla de Enrutamiento para IPv4 e IPv6, la Tabla de Servicios Comunes para IPv4 e IPv6 y gestionar las interfaces de red del sistema operativo.

El diseño general de la interfaz se puede ver en la Figura 6.1 y se divide en cuatro secciones: (1) un panel izquierdo (QListWidget) con una lista de íconos descriptivos que permite la navegación entre los diversos módulos de la herramienta, (2) un panel derecho (QStackedWidget) que muestra los atributos del módulo seleccionado para su edición, (3) un panel inferior común a todos los módulos que posee botones (QPushButton) para salvar, ejecutar y detener el protocolo de enrutamiento, y (4) una barra de estado (QStatusBar) en la parte inferior de la ventana con mensajes relativos al módulo que se edita y otras informaciones. Las dimensiones de la ventana son de 800 px de ancho por 500 px de alto.

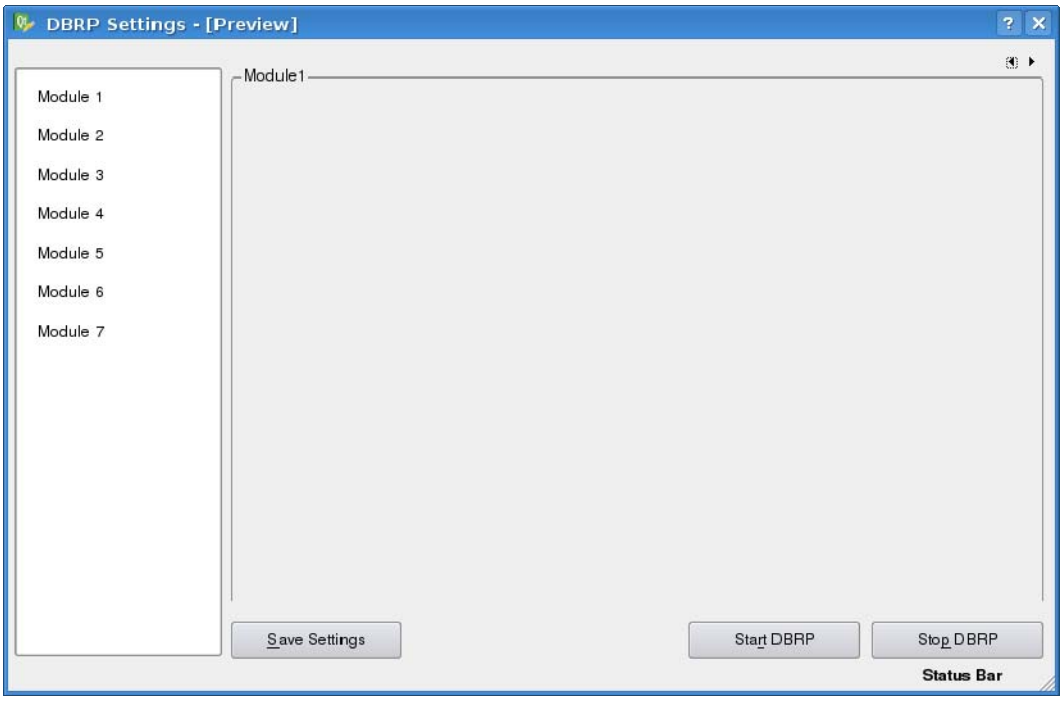

Figura 6.1: Prototipo General de Interfaz

[10] http://www.vmware.com

7. Marco Aplicativo

Este capítulo describirá en base a la metodología XP las iteraciones que fueron necesarias para la implementación de la solución, donde cada iteración muestra su evolución en base a las fases de Diseño, Codificación y Pruebas.

7.1 Análisis General

Como todo servicio en GNU/Linux, el protocolo de enrutamiento DBRP funciona como un proceso Demonio o Daemon (*Disk And Execution MONitor*) que requiere una configuración leída inicialmente de un archivo de texto plano que contiene sus parámetros. Para facilitar la configuración, DBRP posee una herramienta gráfica llamada DBRP Settings que modifica dicho archivo de configuración.

Este análisis permitió determinar de manera global los principales requerimientos de la aplicación cubriendo los objetivos definidos en el Capítulo 2. A continuación se muestra cómo fue planteada la lista de requerimientos para generar tanto la implementación del protocolo DBRP como su herramienta de administración DBRP Settings:

- Desarrollo de procedimientos que permitan el intercambio de mensajes entre PCs, a nivel de capa de enlace según el modelo de referencia OSI, utilizando RAW Sockets.
- Diseño de estructuras para manipular PDUs y TLVs.
- Diseño del archivo de configuración en texto plano que es la entrada para el proceso Demonio, definición del método para su lectura y almacenamiento en memoria.
- Procesamiento de información de enrutamiento.
- Procesamiento de información de servicios comunes.
- Implementación de autenticación y cifrado de los mensajes.
- Desarrollo de la herramienta de configuración gráfica DBRP Settings en Qt, basada en el prototipo general de interfaz planteado en el Marco Metodológico.
- Interacción entre proceso Demonio y DBRP Settings.

7.2 Desarrollo de la Aplicación

La aplicación consta de dos partes, el protocolo de enrutamiento DBRP y la herramienta de configuración DBRP Settings. Las iteraciones del 1 al 5 explican con detalle el desarrollo del protocolo de enrutamiento, la iteración 6 corresponde al desarrollo de DBRP Settings y por último la iteración 7 explica cómo interactúan entre sí ambas partes.

Es importante resaltar que el desarrollo del demonio DBRP está basado en programación estructurada, explotando la capacidad híbrida que posee el lenguaje C++, mientras que el desarrollo de la interfaz gráfica DBRP Settings está basada en programación orientada a objetos, haciendo uso de las bibliotecas que provee Qt para tal fin.

A continuación se especifican cada una de las iteraciones que fueron necesarias y las fases implementadas en ellas según el modelo de programación XP.

7.2.1 Iteración 1: Diseño y Lectura del Archivo de Configuración

Archivo de texto plano con nombre *dbrp.conf* ubicado en el directorio raíz de DBRP.

- **Fase de Diseño:** está compuesto por etiquetas para diferenciar los atributos de cada sección, cada etiqueta está seguida por líneas con pares Parámetro-Valor separados por el símbolo igual (“=”), la cantidad de dichos pares está determinada por el tipo de etiqueta. Este archivo de configuración es generado con la sintaxis adecuada a través de DBRP Settings que será descrito en la Iteración 5 (ver Subsección 7.2.5). Cada etiqueta y sus parámetros son leídos secuencialmente por línea.

  Las secciones del archivo son las siguientes:

  - *Routing Settings:* almacena los parámetros Campo delay: retardo en nano segundos, introducido por el router con valor numérico en el rango de 0 – 16.777.215 (2^{24}-1). Campo domain_id: identificador del dominio de enrutamiento, con valor numérico entre 0 – 255. Debe estar presente sólo una vez en el archivo, sin importar su posición dentro del mismo (ver Figura 7.1).

    ```
    ROUTING_SETTINGS
    delay = 50
    domain_id = 255
    ```

    Figura 7.1: Etiqueta Routing Settings

  - *Interface*: almacena los parámetros de una interfaz de red, nombre de interfaz (*ifname*) con valor alfanumérico. Enrutamiento IPv4 (*ipv4_enable*), con valor numérico 1 para indicar que enruta IPv4 y 0 en caso contrario. Enrutamiento IPv6 (*ipv6_enable*), con valor numérico 1 para indicar que enruta IPv6 y 0 en caso contrario. Al menos una pila de protocolo IP debe tener valor 1 para que DBRP enrute en esta interfaz. Retardo de Interfaz (*ifdelay*), con valor numérico en el rango 0 – 16.777.215. Interfaz DBRP (*ifdbrp*), con valor numérico 1 para indicar que es capaz de enrutar con el protocolo DBRP y 0 en caso contrario (ver Figura 7.2). Pueden haber múltiples etiquetas de este tipo en cualquier orden y posición. En caso de que exista alguna etiqueta cuyo nombre de interfaz no corresponda con

ninguna interfaz en el sistema operativo entonces será ignorada. Para que cada PC pueda enrutar debe tener al menos dos interfaces de red que estén configuradas con DBRP.

```
INTERFACE
ifname = eth0
ipv4_enable = 1
ipv6_enable = 0
ifdelay = 10
ifdbrp = 1
```

Figura 7.2: Etiqueta Interface

- *DNS IPv4 Services*: almacena los parámetros de servicios comunes. DNS IPv4 (*dns4*), con valor numérico 1 si es transportado este servicio y 0 en caso contrario. Prefijo DNS IPv4 (*dns4_prefix*), con valor de dirección IPv4 del prefijo al que está dirigido. Máscara de Subred (*dns4_mask*), con valor numérico en notación CIDR en el rango 0 - 32. Prioridad (*dns4_prior*), con valor numérico en el rango 0 – 65.535. Lista de Direcciones (*dns4_list*), lista de direcciones IPv4 de servidores DNS separados por el símbolo (“^”) con mínimo de una (1) y un máximo de cinco (5) direcciones (ver Figura 7.3). Puede haber hasta un máximo de diez (10) etiquetas de este tipo en el archivo.

```
DNS4_SERVICES
dns4 = 1
dns4_prefix = 10.0.1.0
dns4_mask = 25
dns4_prior = 50
dns4_list = 10.0.0.254^10.0.1.100
```

Figura 7.3: Etiqueta DNS IPv4 Services

- *DNS IPv6 Services*: almacena los parámetros de servicios comunes. DNS IPv6 (*dns6*), con valor numérico 1 si es transportado este servicio y 0 en caso contrario. Prefijo DNS IPv6 (*dns6_prefix*), con valor de dirección IPv6 del prefijo al que está dirigido. Longitud de Prefijo (*dns6_mask*), con valor numérico en notación CIDR en el rango 0 - 128. Prioridad (*dns6_prior*), con valor numérico en el rango 0 – 65.535. Lista de Direcciones (*dns6_list*), lista de direcciones IPv6 de servidores DNS separados por el símbolo (“^”) con mínimo de una (1) y un máximo de cinco (5) direcciones (ver Figura 7.4). Puede haber hasta un máximo de diez (10) etiquetas de este tipo en el archivo.

```
DNS6_SERVICES
dns6 = 1
dns6_prefix = 2001:db8:5:3::
dns6_mask = 64
dns6_prior = 50
dns6_list = 2001:db8:5:4::500
```

Figura 7.4: Etiqueta DNS IPv6 Services

- *Clear Text Authentication:* almacena los parámetros de autenticación de texto en claro. Campo *clr_txt:* con valor numérico 1 si está habilitado este método y 0 en caso contrario. Campo *key_list*: lista de claves separadas por el símbolo (“^”) con un mínimo de una (1) y un máximo de diez (10) claves de hasta 64 caracteres alfanuméricos (ver Figura 7.5). Debe estar presente sólo una vez en el archivo.

- *MD5 Authentication:* almacena los parámetros de autenticación MD5. Campo *md5*: con valor numérico 1 si está habilitado este método y 0 en caso contrario. Campo *key_list:* lista de claves separadas por el símbolo (“^”) con un mínimo de una (1) y un máximo de diez (10) claves de hasta 64 caracteres alfanuméricos (ver Figura 7.5). Debe estar presente sólo una vez en el archivo.

- *SHA1 Authentication:* almacena los parámetros de autenticación SHA1. Campo *sha1*: con valor numérico 1 si está habilitado este método y 0 en caso contrario. Campo *key_list:* lista de claves separadas por el símbolo (“^”) con un mínimo de una (1) y un máximo de diez (10) claves de hasta 64 caracteres alfanuméricos (ver Figura 7.5). Debe estar presente sólo una vez en el archivo.

  Sólo uno de estos tres métodos de autenticación puede estar habilitado a la vez.

```
CLEAR_AUTH
clr_txt = 0
key_list = key1^key2^key3

MD5_AUTH
md5 = 0
key_list = key1^key2^key3^key4

SHA1_AUTH
sha1 = 1
key_list = key1^key2^key3
```

Figura 7.5: Etiquetas Authentication

- *AES Encryption: almacena los parámetros de cifrado AES. Campo aes*: *con valor numérico 1 si está* habilitado este método y 0 en caso contrario. Campo *key_list:* lista de claves separadas por el símbolo (“^”) con un mínimo de una (1) y un máximo de diez (10) claves de exactamente 16 caracteres alfanuméricos (ver Figura 7.6). Debe estar presente sólo una vez en el archivo.

- *DES Encryption:* almacena los parámetros de cifrado DES. Campo *des*: con valor numérico 1 si está habilitado este método y 0 en caso contrario. Campo *key_list:* lista de claves separadas por el símbolo (“^”) con un mínimo de una (1) y un máximo de diez (10) claves de exactamente 8 caracteres alfanuméricos (ver Figura 7.6). Debe estar presente sólo una vez en el archivo.

- *3DES Encryption:* almacena los parámetros de cifrado 3DES. Campo *3des*: con valor numérico 1 si está habilitado este método y 0 en caso contrario. Campo *key_list:* lista de claves separadas por el símbolo (“^”), deben ser tres (3) claves de exactamente 8 caracteres alfanuméricos (ver Figura 7.6). Debe estar presente sólo una vez en el archivo.

  Sólo uno de estos tres métodos de cifrado puede estar habilitado a la vez.

```
AES_ENCRY
aes = 0
key_list = <key1>^<key2>^<key3>

DES_ENCRY
des = 0
key_list = <key1>^<key2>^<key3>^<key4>

TRIPLE_DES_ENCRY
3des = 1
key_list = <key1>^<key2>^<key3>
```

Figura 7.6: Etiquetas Encryption

- **Fase de Codificación:** en la fase anterior se describió la estructura del archivo de configuración. La función *read_file()* lee secuencialmente el archivo hasta encontrar una etiqueta y compararla con un alfabeto de palabras reservadas definidas en una estructura (ver Figura 7.7). Luego, la etiqueta es procesada por la función *get_parameters()* que lee los parámetros que le siguen, cada uno de estos es comparado con otro alfabeto definido en otra estructura (ver Figura 7.9). Más adelante, la función *get_value()* se encarga de obtener el o los valores de cada parámetro y almacenarlos en su respectiva sección dentro la estructura de configuración (ver Figura 7.8).

```
struct  para_lab  label_abc[ LABEL_SIZE ] =
{
        {"INTERFACE"},          {"CLEAR_AUTH"},         {"MD5_AUTH"},
        {"SHA1_AUTH"},          {"AES_ENCRY"},          {"DES_ENCRY"},
        {"TRIPLE_DES_ENCRY"},   {"ROUTING_SETTINGS"},   {"DNS4_SERVICES"},
        {"DNS6_SERVICES"}
};
```

Figura 7.7: Estructura con Definición de Etiquetas

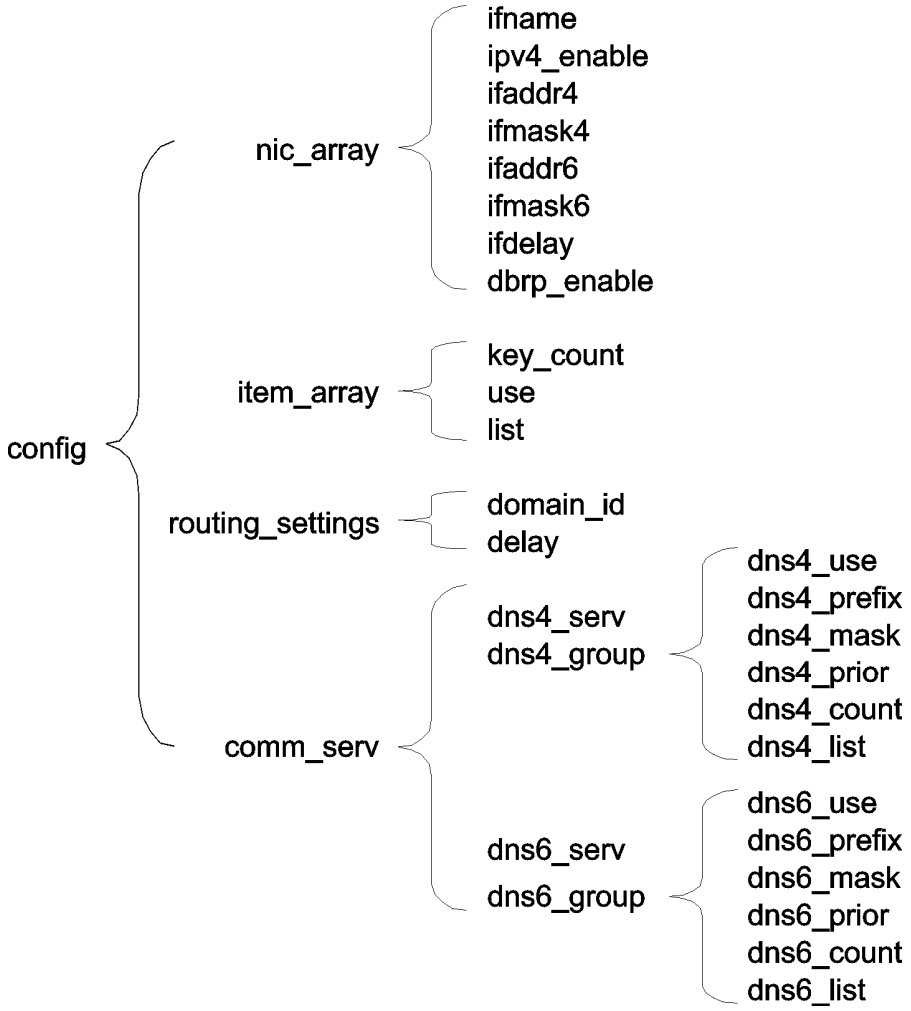

Figura 7.8: Diagrama de Estructura de Configuración

```
struct  para_lab  para_abc[ PARA_SIZE ] =
{
        {"ifname"},       {"clr_txt"},       {"md5"},          {"sha1"},
        {"aes"},          {"des"},           {"3des"},         {"key_list"},
        {"delay"},        {"domain_id"},     {"dns4"},         {"dns4_prefix"},
        {"dns4_mask"},    {"dns4_prior"},    {"dns4_list"},    {"dns6"},
        {"dns6_prefix"},  {"dns6_mask"},     {"dns6_prior"},   {"dns6_list"},
        {"ifdelay"},      {"ifdbrp"},        {"ipv4_enable"},  {"ipv6_enable"}
};
```

Figura 7.9: Estructura con Definición de Atributos

- **Fase de Pruebas:** estas pruebas consistieron en verificar que cada una de las etiquetas y parámetros presentes en el archivo fuesen leídos adecuadamente, contemplando posibles errores de sintaxis en la estructura del archivo, tales como símbolos o separadores no definidos que eventualmente hayan sido incorporados por el administrador de forma manual. Sin embargo, el procedimiento adecuado implica que el administrador debe ejecutar la herramienta DBRP Settings para generar adecuadamente el archivo de configuración deseado evitando así posibles errores.

  Las etiquetas seguidas del conjunto de parámetros que les corresponden, fueron ubicadas en distinto orden dentro del archivo para asegurar que fuesen leídas en todo momento. Fueron tomados en cuenta algunos casos en los que se trata de iniciar el protocolo sin haber generado un archivo de configuración correcto.

  Fue evaluada la adecuada carga en memoria de los valores que son leídos secuencialmente del archivo de configuración.

7.2.2 Iteración 2: Envío y Recepción de Mensajes

Haciendo uso de la comunicación entre procesos a nivel de sockets que provee la biblioteca Sockets de C++, fueron desarrollados procedimientos utilizando RAW sockets que permiten el intercambio de mensajes entre PCs a nivel de capa de enlace según el modelo de referencia OSI.

- **Fase de Diseño:** el encapsulamiento de un mensaje DBRP dentro de una trama Ethernet se muestra en la Figura 7.10.

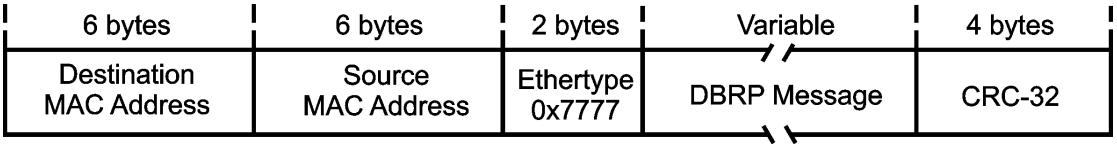

Figura 7.10: Ejemplo de un Mensaje DBRP en una Trama Ethernet

Para establecer roles de envío y recepción fue necesaria la implementación de los hilos *client* y *daemon*. El hilo *client* se encarga de enviar mensajes Request, Update y Shutdown de acuerdo a tareas dejadas por el hilo *daemon.* El hilo *daemon* se encarga de recibir todos los mensajes Ethernet filtrando por el campo Ethertype definido para DBRP. Una vez recibido un mensaje y verificada la integridad del mismo, el hilo *daemon* deja una tarea con el contenido del mensaje al hilo *client* para su respectivo procesamiento y respuesta de acuerdo al tipo de mensaje. Para cada interfaz de red que enrute DBRP se crean un hilo *client* y *daemon* para enviar y recibir mensajes respectivamente.

El campo Destination MAC Address de la trama Ethernet es una dirección MAC *multicast* con valor 01:00:00:77:77:77 para mensajes de tipo Request y Shutdown. Para los mensajes de tipo Update se usa la misma dirección Multicast en caso de ser un Update regular (síncrono), y cuando sea un Update no regular (asíncrono) en respuesta a un Request, es usada la dirección *unicast* del PC solicitante. El campo Source MAC Address siempre corresponde a la dirección de la interfaz de red que envía el mensaje. El campo Ethertype es 0x7777, el campo DBRP Message contiene el PDU DBRP. El campo CRC-32 lleva el hash de redundancia cíclica. Para poder recibir mensajes con la dirección destino *multicast*, es necesario que cada interfaz de red de cada PC se suscriba al grupo *multicast* correspondiente, esta tarea es hecha por el protocolo durante su ejecución.

- **Fase de Codificación:** la comunicación entre PCs se hace a nivel de capa de enlace, para lo cual es necesario crear un socket de envío y otro para recepción en cada PC. Este tipo de sockets requieren parámetros específicos en su inicialización mediante la función *socket()* de C++ (ver Figura 7.11).

```
int socket( int domain, int type, int protocol )
```

Figura 7.11: Función socket

Donde *domain* especifica el dominio de comunicación en el cual el socket será creado, para el intercambio de paquetes en capa de enlace este parámetro debe ser PF_PACKET, *type* indica el tipo de socket y será SOCK_RAW, por último *protocol* indica el protocolo soportado para la familia de direcciones, en este caso ETH_P_ALL. Esta función devuelve en caso exitoso un valor entero positivo del descriptor de archivo.

Para el envío se usa la función descrita en la Figura 7.12, cuyos parámetros son: *sock* es el descriptor de archivo devuelto por la función *socket()*, *msg* es el mensaje que será enviado, *len* la longitud del mensaje, *flags* con valor MSG_DONTROUTE usado para indicar que el mensaje no debe ser enrutado en el destino, *addr* indica la estructura que contiene la dirección de destino, *alen* indica la longitud de la estructura *addr*.

```
int sendto( int sock, const void *msg, size_t len, int flags, const struct sockaddr *addr, socklen_t alen )
```

Figura 7.12: Función sendto

Para la recepción se usa la función descrita en la Figura 7.13, cuyos parámetros son iguales a la función *sendto()* salvo que el valor de *msg* es donde será almacenado el mensaje y *flags* debe ser MSG_TRUNC para que la función devuelva la longitud real del mensaje recibido.

```
Int recvfrom( int socket, void *restrict msg, size_t len, int flags,
                struct sockaddr *restrict address, socklen_t *restrict address_len )
```

Figura 7.13: Función recvfrom

- **Fase de Pruebas:** en las pruebas unitarias para envío y recepción de mensajes, se utilizó la herramienta de análisis de protocolos Wireshark con el complemento para DBRP, con la cual fue posible visualizar y depurar cualquier error o malformación de mensajes. Se pueden observar el intercambio de mensajes DBRP entre dos PCs empleando autenticación SHA1 y pilas IPv4 e IPv6. PDU Request (ver Figura 7.14), PDU Update (Triggered, ver Figura 7.15), PDU Update (Unicast, ver Figura 7.16) y PDU Shutdown (ver Figura 7.17).

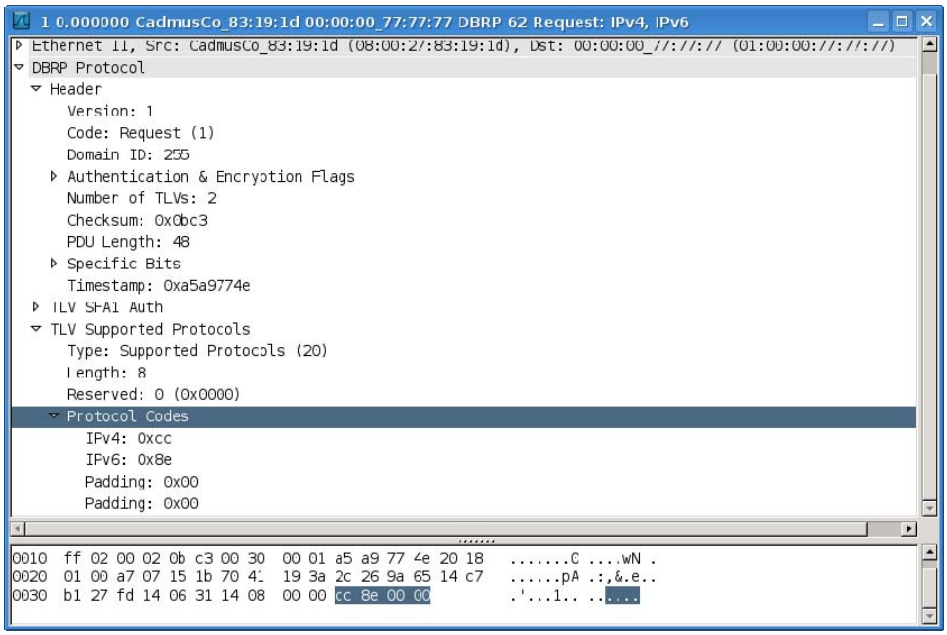

Figura 7.14: Captura de PDU Request

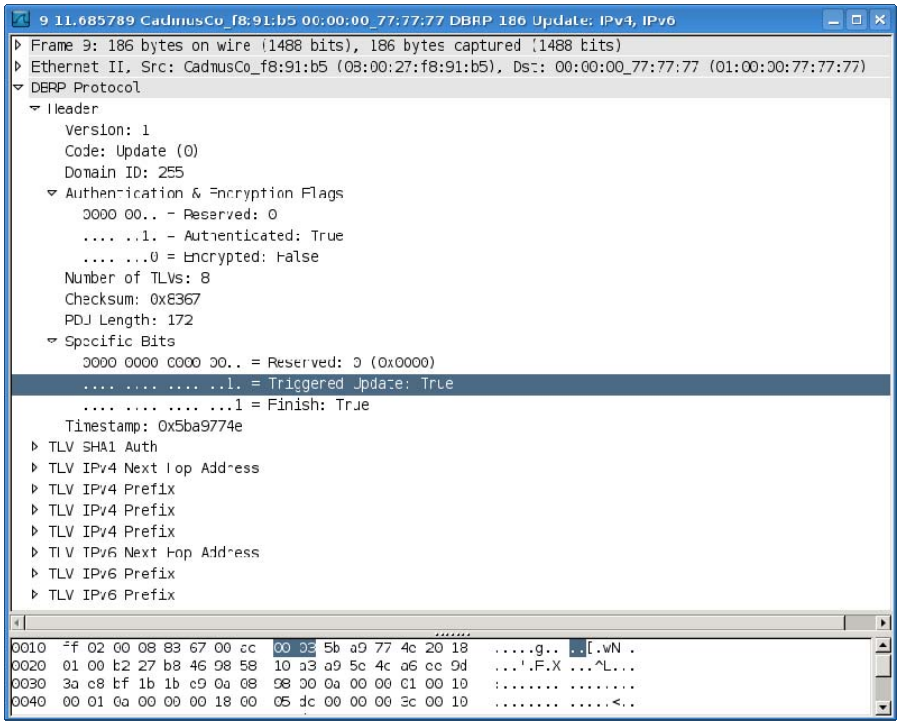

Figura 7.15: Captura de PDU Update (Triggered)

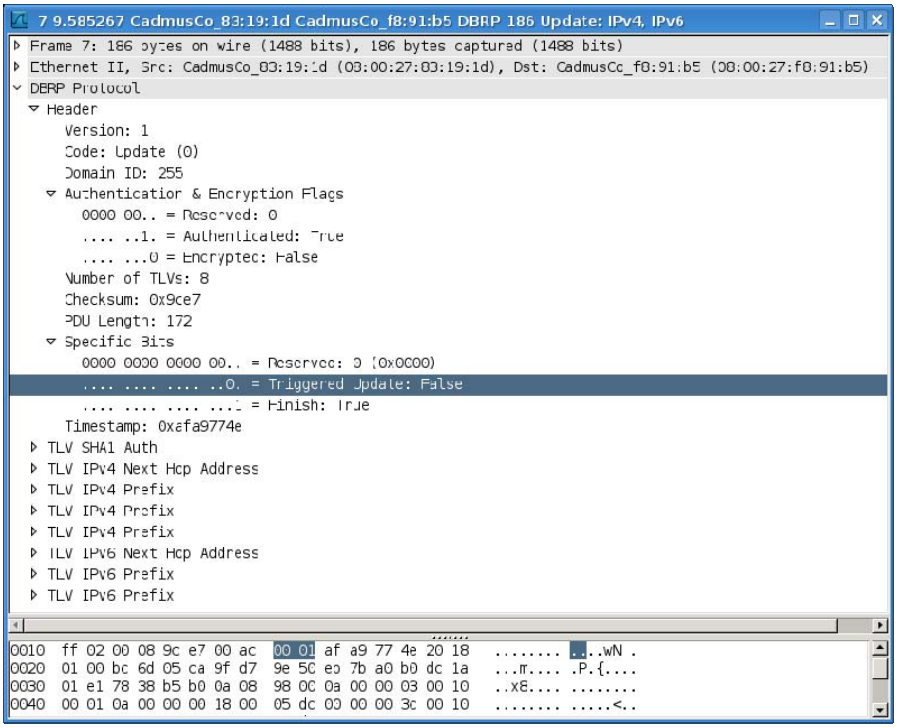

Figura 7.16: Captura de PDU Update (Unicast)

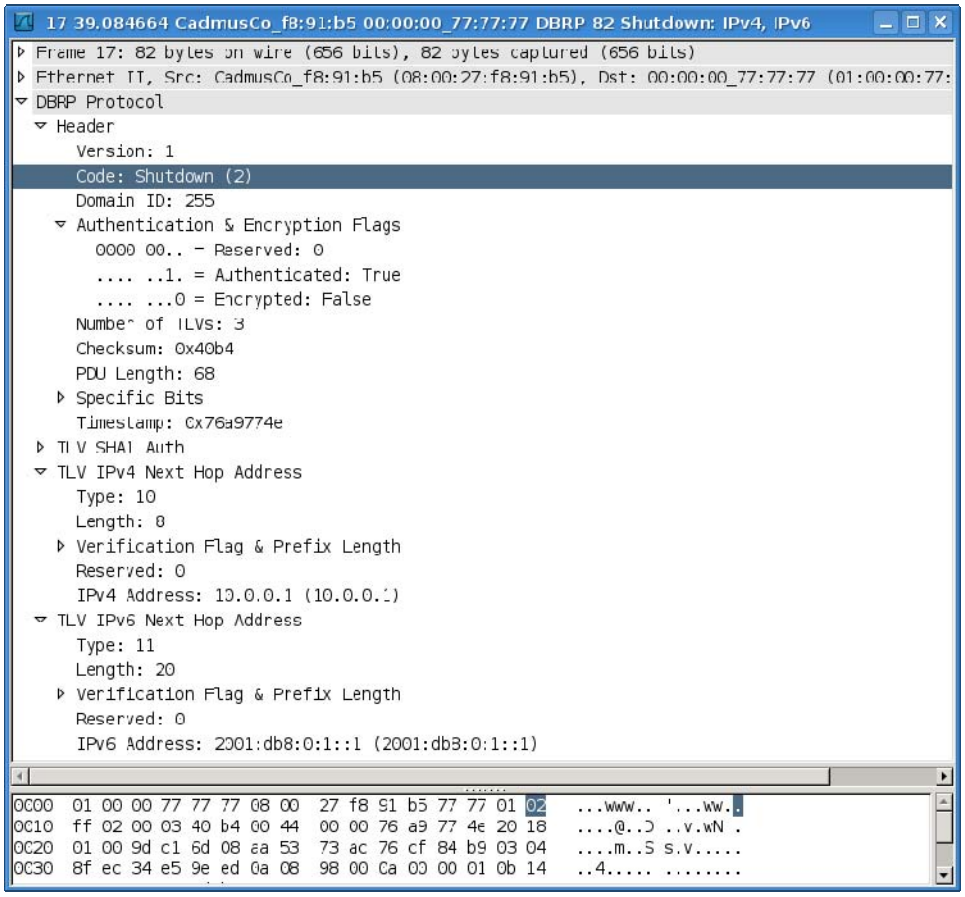

Figura 7.17: Captura de PDU Shutdown

7.2.3 Iteración 3: Implementación de Autenticación y Cifrado

Con el objetivo de ofrecer un protocolo de enrutamiento seguro, esta versión de DBRP (v1) implementa tres algoritmos de autenticación (Clear Text, MD5 y SHA1) y tres algoritmos de cifrado (AES, DES y 3DES) proponiendo tres escenarios de seguridad: (1) sin autenticación ni cifrado recomendado sólo para pruebas, (2) con autenticación sin cifrado y (3) con autenticación y cifrado, siendo posible combinar un algoritmo de autenticación con uno de cifrado a conveniencia.

- **Fase de Diseño:** de acuerdo a la configuración que establece el administrador, el protocolo aplica los mecanismos de seguridad siguientes:

  - Cuando se envía un mensaje y la autenticación está habilitada, cada PDU (*Request*, *Update* y *Shutdown*) contiene el TLV de autenticación de acuerdo al algoritmo seleccionado y sus campos deben ser calculados luego de ensamblar el mensaje completo. Si adicionalmente el cifrado está habilitado, se cifra sólo la porción sensible del PDU *Update* ya que es el único que contiene información de enrutamiento y servicios comunes, y por último se calculan los campos del TLV de autenticación.

- Cuando se recibe un mensaje y la autenticación está habilitada, se verifica su autenticidad comparando la clave configurada por el administrador según el índice que viaja en el TLV de autenticación (esto para el caso de Clear Text) o se compara el *hash* resultante de aplicar el algoritmo (para los casos de MD5 y SHA1), luego se procesa el mensaje. Si adicionalmente el cifrado está habilitado se descifra con el índice de algoritmo y clave indicados en el TLV de cifrado y luego es efectuado su respectivo procesamiento.

- **Fase de Codificación:** los mecanismos de seguridad que posee DBRP fueron implementados haciendo uso de primitivas de la biblioteca OpenSSL para autenticación y cifrado.

  Para el caso de la autenticación Clear Text la información de la clave en el TLV viaja en claro y el receptor compara esta clave con la que tiene definida según el Key ID. De ser iguales se prosigue con el procesamiento de los demás TLVs del PDU, de lo contrario es descartado el mensaje.

  En la autenticación MD5, se escoge aleatoriamente la clave de la lista de claves y su índice es almacenado en el TLV *MD5 Authentication* en el campo Key ID. Esta clave se concatena con el *buffer* de datos. Luego es necesario inicializar variables de contexto requeridas por la función *MD5_Init()*, se pasa la variable de contexto, el *buffer* de datos y su longitud a la función *MD5_Update()*, por último la función *MD5_Final()* hace el cálculo del *hash*, que finalmente es almacenado en el TLV. Para verificar la autenticidad en el receptor, se guarda una copia del *hash* recibido, se asigna en ceros el original y se repite el procedimiento anterior con la clave que indique el Key ID, luego el *hash* resultante es comparado con la copia y deben ser iguales, en caso contrario es descartado el mensaje.

  En la autenticación SHA1, se escoge aleatoriamente la clave de la lista de claves y su índice es almacenado en el TLV *SHA1 Authentication* en el campo Key ID. Esta clave se concatena con el *buffer* de datos. Luego es necesario inicializar variables de contexto requeridas por la función *SHA1_Init()*, se pasa la variable de contexto, el *buffer* de datos y su longitud a la función *SHA1_Update()*, por último la función *SHA1_Final()* hace el cálculo del *hash*, que finalmente es almacenado en el TLV. Para verificar la autenticidad en el receptor, se guarda una copia del *hash* recibido, se asigna en ceros el original y se repite el procedimiento anterior con la clave que indique el Key ID, luego el *hash* resultante es comparado con la copia y deben ser iguales, en caso contrario es descartado el mensaje.

  Para cifrar con AES, DES ó 3DES, se escoge aleatoriamente la clave de la lista de claves del algoritmo y su índice es almacenado en el TLV *Encryption* en el campo Key ID, se asigna el ID correspondiente en el campo Algorithm ID como se indica en la Tabla 5.16, se inicializan variables de contexto, es llamada la función *EVP_EncryptInit_ex()* cuyos parámetros a su vez pueden ser las funciones *EVP_aes_128_cbc(), EVP_des_cbc()* ó *EVP_des_ede3_cbc()*

dependiendo del tipo de cifrado, por último la función *EVP_EncryptUpdate()* retorna el *buffer* con el contenido cifrado y su longitud final. Para descifrar se sigue el procedimiento anterior con la clave y algoritmo indicados pero sustituyendo las funciones *EVP_EncryptInit_ex()* y *EVP_EncryptUpdate()* por *EVP_DecryptInit_ex()* y *EVP_DecryptUpdate()*.

- **Fase de Pruebas:** pruebas unitarias para cada tipo de autenticación así como cada tipo de cifrado, permitieron verificar la correctitud en la aplicación de los algoritmos correspondientes.

  Para comprobar los métodos de autenticación, se efectuaron múltiples pruebas con claves distintas en dos o más routers, obteniendo que los mensajes eran descartados, al modificar las claves para que sean iguales, la comunicación es exitosa y los routers convergen correctamente.

  En el caso del cifrado, fue necesario analizar los pasos indicados por cada algoritmo y aplicar las primitivas adecuadas, haciendo llamadas a las funciones de cifrado y descifrado múltiples veces para verificar la integridad y correctitud de los datos. Al principio se efectuaron pruebas con datos aleatorios y de distintos tamaños, para finalmente probar con los PDUs del protocolo.

  Por ejemplo, haciendo capturas de tramas con la herramienta de análisis de protocolos Wireshark fue posible visualizar que el contenido de los datos en el PDU viajase autenticado (Clear Text) y cifrado (AES) cuando están habilitados estos métodos (ver Figura 7.18).

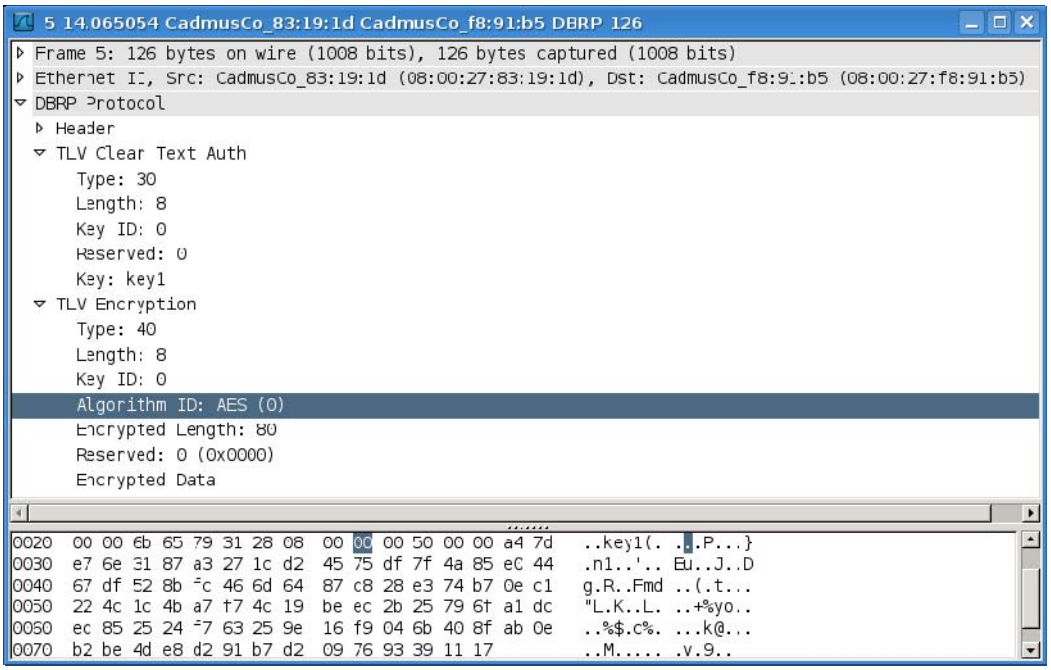

Figura 7.18: Trama con Autenticación y Cifrado

7.2.4 Iteración 4: Procesamiento de Información de Enrutamiento y Servicios Comunes

El objetivo principal de un protocolo de enrutamiento es mantener su tabla de enrutamiento actualizada y con información coherente. Para lograr esto es necesario el intercambio de mensajes entre los routers. Este intercambio se lleva a cabo con el envío y recepción de mensajes *Update* que contienen la información de los prefijos alcanzables y los cambios que se susciten en todo momento así como la información referente a servicios comunes.

- **Fase de Diseño:** DBRP mantiene su propia TE (*Tabla de Enrutamiento*) en memoria. Es actualizada cuando se presentan cambios y estos se replican en la TE del sistema operativo. La TE en memoria está conformada por cuatro listas enlazadas, cada lista esta respectivamente compuesta por nodos de prefijos IPv4 (ver Figura 7.20), prefijos IPv6 (ver Figura 7.21), DNS IPv4 (ver Figura 7.22) y DNS IPv6 (ver Figura 7.23). La estructura de la TE se puede detallar en la Figura 7.19. En las variables *modified* y *triggered_update* se registra si se ha modificado recientemente y si es necesario enviar un *Triggered Update* respectivamente.

```
struct routing_table {
        struct ipv4_prefix      *ipv4_first;
        struct ipv4_prefix      *ipv4_last;
        int                     ipv4_count;
        struct ipv6_prefix      *ipv6_first;
        struct ipv6_prefix      *ipv6_last;
        int                     ipv6_count;

        struct ipv4_dns         *dns4_first;
        struct ipv4_dns         *dns4_last;
        int                     dns4_count;
        struct ipv6_dns         *dns6_first;
        struct ipv6_dns         *dns6_last;
        int                     dns6_count;

        int                     modified;
        int                     triggered_update;
}
```

Figura 7.19: Estructura de Tabla de Enrutamiento

```
struct ipv4_prefix {
        u_int8_t        type;
        u_int8_t        length;
        u_int16_t       res_d_i;
        struct in_addr  prefx_id;
        u_int8_t        prefx_len;
        u_int8_t        hop_count;
        u_int16_t       min_mtu;
        u_int32_t       sum_delay;

        struct in_addr  next_hop;
        char            if_name[16];
        int             holddown;
        int             dead_interval;
        int             modified;
        int             direct;
        int             triggered;
        char            cmd[10];

        struct ipv4_prefix *next;
}
```

Figura 7.20: Estructura Utilizada en las Listas de IPv4 Prefix

```
struct ipv6_prefix {
        u_int8_t        type;
        u_int8_t        length;
        u_int16_t       res_d_i;
        struct in6_addr prefx_id;
        u_int8_t        prefx_len;
        u_int8_t        hop_count;
        u_int16_t       min_mtu;
        u_int32_t       sum_delay;

        struct in6_addr next_hop;
        char            if_name[16];
        int             holddown;
        int             dead_interval;
        int             modified;
        int             direct;
        int             triggered;
        char            cmd[10];
        struct ipv6_prefix *next;
}
```

Figura 7.21: Estructura Utilizada en las Listas de IPv6 Prefix

```
struct ipv4_dns {
        u_int8_t        type;
        u_int8_t        length;
        u_int16_t       res_d_i;
        struct in_addr  prefx_id;
        u_int8_t        prefx_len;
        u_int8_t        numb_dns;
        u_int16_t       prior;
        struct in_addr  dns_addr[5];

        struct ipv4_dns *next;
        char            if_name[16];
        int             modified;
        int             direct;
        int             dead_interval;
}
```

Figura 7.22: Estructura Utilizada en las Listas de DNS IPv4

```
struct ipv6_dns {
        u_int8_t        type;
        u_int8_t        length;
        u_int16_t       res_d_i;
        struct in6_addr prefx_id;
        u_int8_t        prefx_len;
        u_int8_t        numb_dns;
        u_int16_t       prior;
        struct in6_addr dns_addr[5];

        struct ipv6_dns *next;
        char            if_name[16];
        int             modified;
        int             direct;
        int             dead_interval;
}
```

Figura 7.23: Estructura Utilizada en las Listas de DNS IPv6

- **Fase de Codificación:** la manipulación de la TE en memoria se hace con las primitivas *insert(), delete(), modify()* y *find()* con las cuales es posible insertar, borrar, modificar y encontrar prefijos en esta tabla.

  Cuando el protocolo inicia, son insertados en la TE los servicios comunes y los prefijos directamente conectados de las interfaces de red configuradas para DBRP. Luego se envían mensajes *Request* por estas interfaces para solicitar información de enrutamiento a los routers vecinos de los protocolos soportados.

Un router que recibe un *Request* verifica los protocolos que soporta y los compara con los solicitados, si alguno o todos coinciden entonces debe responder con un *Update* que contiene los prefijos de dichos protocolos.

Un router que recibe un *Update* verifica el campo Number of TLVs de la cabecera del PDU para conocer cuánto debe iterar sobre éste. En cada iteración se verifica el tipo de TLV y su longitud para saber cómo tratarlo y desplazarse al próximo TLV.

Para el caso de un TLV IPv4 Next Hop Address se consulta la bandera V (*Verification*) la cual indica que el router receptor del PDU debe verificar que comparte la misma red del router emisor. Si V tiene valor 1 se debe enmascarar la dirección IPv4 contenida en el campo IPv4 Address con la longitud del prefijo Prefix Lenght para obtener la dirección de red, la cual se compara con la dirección de red de la interfaz por donde fue recibido el mensaje. Si son iguales se procesan los TLVs IPv4 Prefix que siguen, en caso contrario son descartados.

Cada TLV IPv4 Prefix es previamente buscado en la TE, si la bandera I (*Insert*) se encuentra activa y el prefijo no existe en la TE entonces se inserta, de lo contrario se verifican tres situaciones:

- Mejor métrica: se actualiza la métrica, se marca el prefijo como modificado y se reinician las banderas *Dead Interval* y *Holddown* del prefijo.
- Igual métrica: se reinician las banderas *Dead Interval* y *Holddown* del prefijo.
- Peor métrica: si el prefijo es anunciado por el *Next Hop*, se aplica el *Holddown* ignorando este anuncio durante dos veces el *Update Interval*. Si el prefijo es anunciado por otro router es descartado.

Si la bandera D (*Delete*) se encuentra activa y el prefijo existe en la TE entonces se marca para eliminación, y luego de ser anunciado este cambio en un *Triggered Update* se elimina de la TE. Si no existe en la TE se ignora el anuncio de borrado de este prefijo.

Para el caso de un TLV IPv6 Next Hop Address no se consulta la bandera V, ya que el campo IPv6 Address siempre es una dirección Link Local. Cada TLV IPv6 Prefix es previamente buscado en la TE, si la bandera I (*Insert*) se encuentra activa y el prefijo no existe en la TE entonces se inserta, de lo contrario se verifican tres situaciones:

- Mejor métrica: se actualiza la métrica, se marca el prefijo como modificado y se reinician las banderas *Dead Interval* y *Holddown* del prefijo.
- Igual métrica: se reinician las banderas *Dead Interval* y *Holddown* del prefijo.
- Peor métrica: si el prefijo es anunciado por el *Next Hop*, se aplica el *Holddown* ignorando este anuncio durante dos veces el *Update Interval*. Si el prefijo es anunciado por otro router entonces es descartado.

Si la bandera D (*Delete*) se encuentra activa y el prefijo existe en la TE entonces se marca para eliminación, y luego de ser anunciado este cambio en un *Triggered Update* se elimina de la TE. Si no existe en la TE se ignora el anuncio de este prefijo.

El administrador puede hacer cambios a través de DBRP Settings que ameriten el envío de un *Shutdown*, estos cambios pueden ser:

- Desactivación de una pila IPv4 o IPv6: cuando esto ocurre es necesario enviar un *Shutdown* que contiene el TLV Next Hop Address por la interfaz asociada al prefijo desactivado, y por las otras interfaces debe ser anunciado un *Update* indicando el cambio.
- Desactivación de una interfaz: cuando esto ocurre es necesario enviar un *Shutdown* que contiene los TLVs Next Hop Address de todas las pilas activas por la interfaz a desactivar, y por las otras interfaces debe ser anunciado un *Update* indicando el cambio.

Para el caso de los servicios comunes los TLVs DNS IPv4 y DNS IPv6 son previamente buscados en la TE, si la bandera I (*Insert*) se encuentra activa y el DNS no existe en la TE entonces se inserta, de lo contrario se verifican tres situaciones:

- Mayor prioridad: se actualiza la prioridad, se marca el DNS como modificado y se reinicia la bandera *Dead Interval* del DNS.
- Igual prioridad: se reinicia la bandera *Dead Interval* del DNS.
- Menor prioridad: siempre es descartado.

Si la bandera D (*Delete*) se encuentra activa y el DNS existe en la TE entonces se marca para eliminación esté dirigido o no a este router. Cuando el DNS no está dirigido al router entonces es anunciado este cambio en un *Triggered Update* y luego se elimina de la TE. Si no existe en la TE se ignora el anuncio de este DNS.

Si el campo Prefix Identifier del TLV DNS corresponde con alguno de los prefijos directamente conectados entonces se entiende que está dirigido a este router y no se propaga, de lo contrario debe ser anunciado por la interfaz con la cual se alcanza al prefijo correspondiente.

Un router que recibe un *Shutdown* debe buscar en la TE y marcar para eliminar todos los prefijos cuyos Next Hop correspondan con la dirección IP de los campos IPv4 Address o IPv6 Address en los TLVs IPv4 Next Hop Address e IPv6 Next Hop Address respectivamente contenidos en el *Shutdown*. Estos cambios son anunciados con un *Triggered Update* y luego son actualizados en la TE.

- **Fase de Pruebas:** para verificar la correctitud de la estructura de la TE en memoria se efectuó una depuración minuciosa de los nodos, los valores de los campos y los enlaces para su adecuado recorrido. Fueron evaluadas con pruebas unitarias cada una de las primitivas para manipulación de la TE.

  Todo cambio efectuado en la TE de memoria fue replicado en la TE del sistema operativo a través de comandos IP de GNU/Linux con los cuales se mantenía la sincronización entre ambas tablas.

7.2.5 Iteración 5: Desarrollo de Interfaz Gráfica

La interfaz gráfica es la aplicación que permite definir los parámetros correctos para la ejecución del protocolo y es llamada DBRP Settings. De acuerdo a los parámetros definidos en el archivo de configuración descrito en la Subsección 7.2.1, esta interfaz cuenta con los siguientes módulos: *Interfaces*, *Routing Settings*, *Authentication*, *Encryption*, *Tables*, *DNS IPv4 Services* y *DNS IPv6 Services*.

- **Fase de Diseño:** tal como se menciona en la Sección 6.3, los distintos módulos pueden ser seleccionados desde el panel izquierdo a través de una lista de íconos descriptivos. Luego de hacer modificaciones sobre cualquier módulo es necesario presionar el botón Save Settings para salvar las mismas, y ello debe hacerse antes de cambiar a otro módulo. En el panel derecho se encuentran los elementos gráficos necesarios para configurar cada módulo, que a continuación serán descritos:

  - Módulo Interfaces: este módulo permite configurar los parámetros de DBRP en las interfaces de red del sistema operativo (ver Figura 7.24). Son posibles las siguientes configuraciones: (1) una vez seleccionada la interfaz de red mediante el elemento *QComboBox* es posible cambiar su estado Up/Down y ver el estado actual con una bandera indicativa junto al nombre de la interfaz, (2) con el uso de *QCheckBox* se pueden habilitar y deshabilitar tanto el protocolo como las pilas IPv4 e IPv6 en esta interfaz, (3) las direcciones IP y sus respectivas máscaras de red son editables mediante *QLineEdit* validando su correcto formato y el posible solapamiento de direcciones entre las restantes interfaces de red, el resultado de esta validación es mostrado al lado derecho de cada campo haciendo uso de imágenes y textos explicativos, y (4) es posible añadir y eliminar interfaces virtuales dummy. Cada uno de estos elementos tiene un acceso directo asociado.

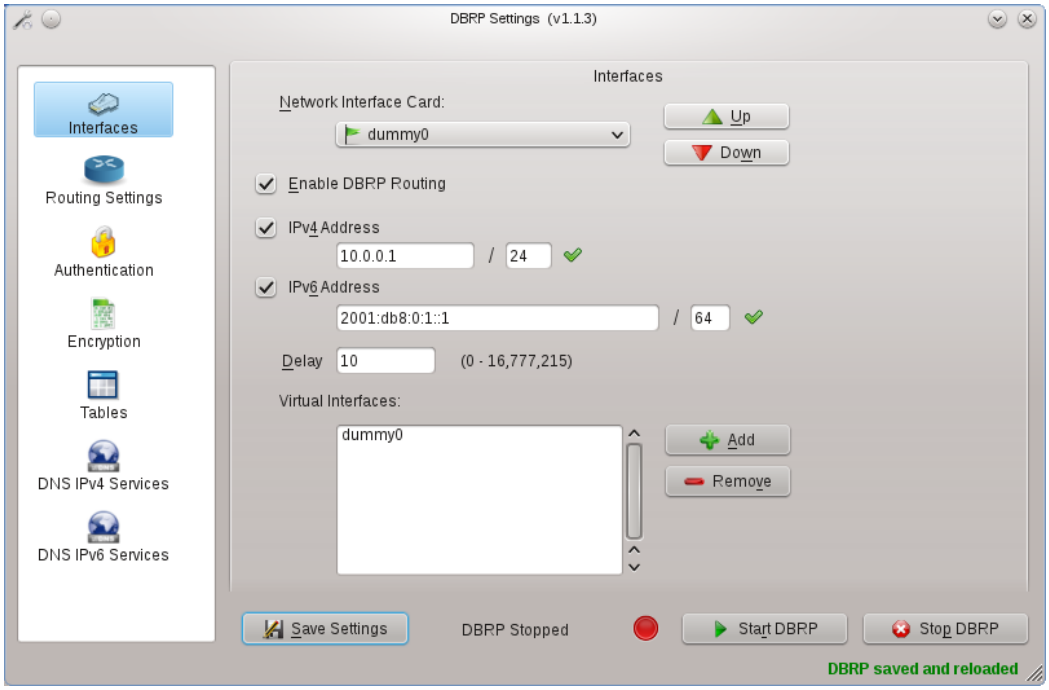

Figura 7.24: Módulo Interfaces de DBRP Settings

- Módulo Routing Settings: permite configurar los parámetros Domain ID y Delay del router con *QLineEdit*, validando los rangos numéricos haciendo uso de *QValidator* (ver Figura 7.25).

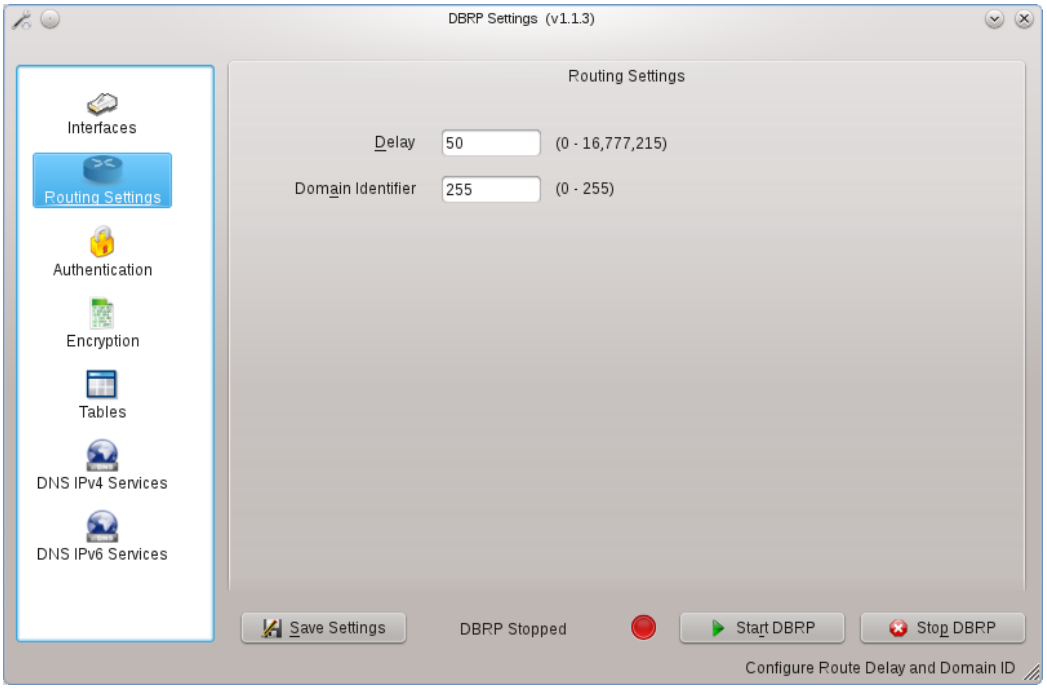

Figura 7.25: Módulo Routing Settings de DBRP Settings

- Módulo Authentication: permite configurar los parámetros de autenticación (ver Figura 7.26). El usuario puede: (1) seleccionar el tipo de autenticación con *QRadioButton*, (2) añadir y eliminar claves con *QLineEdit* y *QButton*, validando la longitud máxima de 64 caracteres, (3) editar una clave ya creada haciendo doble clic sobre ella en la tabla *QTableWidget,* y (4) modificar el orden de las claves creadas con los ascensores del lado derecho de la tabla.

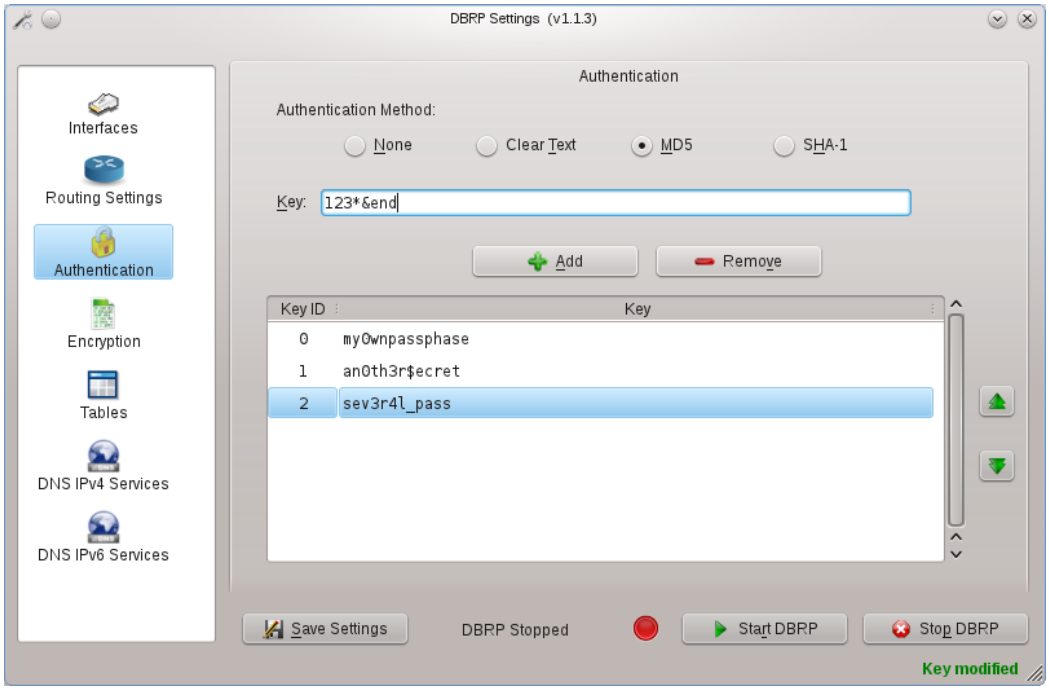

Figura 7.26: Módulo Authentication de DBRP Settings

- Módulo Encryption: permite configurar los parámetros de cifrado (ver Figura 7.27). El usuario puede: (1) seleccionar el tipo de cifrado con *QRadioButton,* (2) añadir y eliminar claves con *QLineEdit* y *QButton*, validando la longitud exacta de 16 caracteres para AES y 8 caracteres para DES y 3DES, (3) editar una clave ya creada haciendo doble clic sobre ella en la tabla *QTableWidget,* y (4) modificar el orden de las claves creadas con los ascensores del lado derecho de la tabla. Para 3DES es necesario crear exactamente tres claves por cada grupo que se desee añadir.

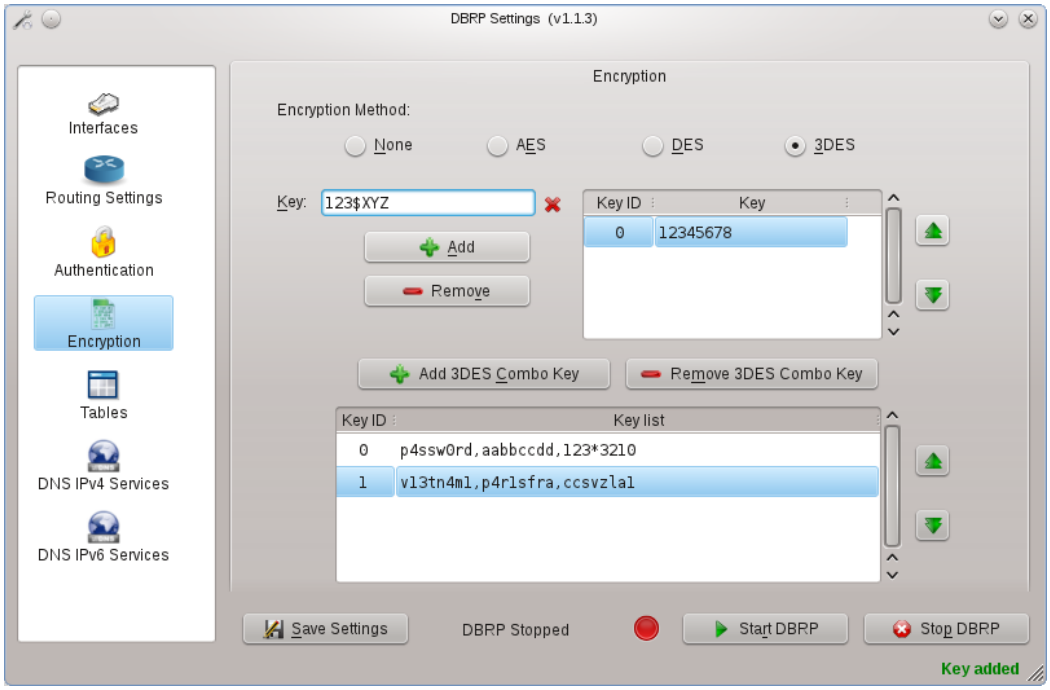

Figura 7.27: Módulo Encryption de DBRP Settings

- Módulo Tables: permite visualizar las Tablas de Enrutamiento de IPv4 e IPv6 y las Tablas de Servicios Comunes DNS IPv4 y DNS IPv6 seleccionando cada una desde el *QComboBox* (ver Figura 7.28). Estas tablas son alimentadas periódicamente por el protocolo en ejecución.

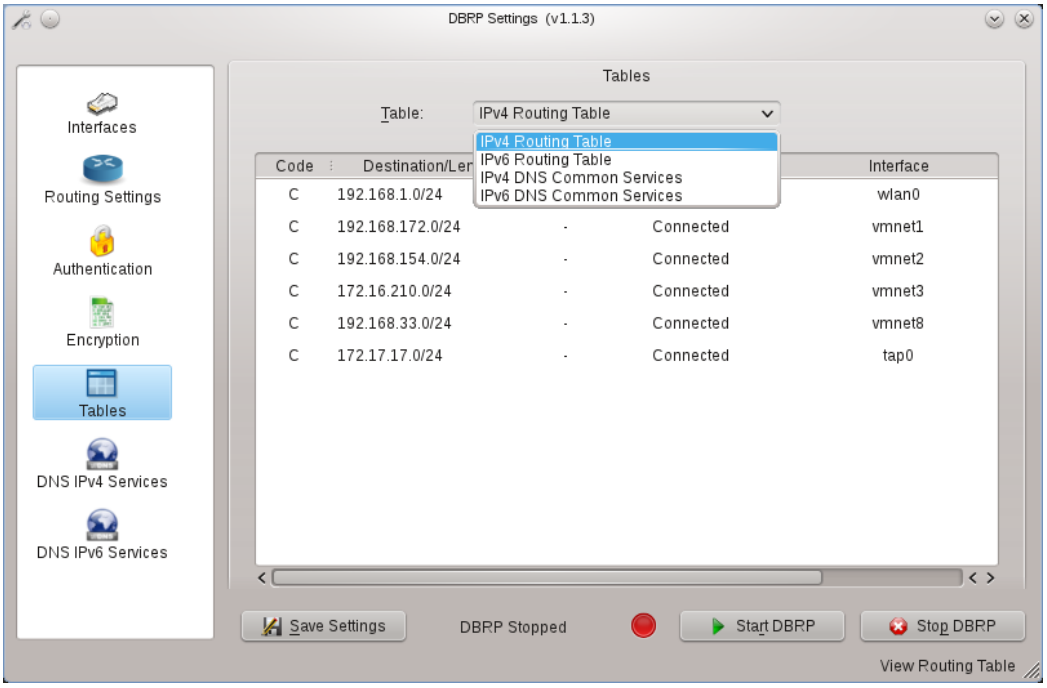

Figura 7.28: Módulo Tables de DBRP Settings

- Módulo DNS IPv4 Services: permite configurar los servidores DNS destinados a determinada red y que serán transportados con DBRP (ver Figura 7.29). El usuario puede: (1) configurar el prefijo de la red destino para el cual están dirigidos los servidores DNS, validando el formato correcto, (2) establecer la prioridad de los servidores DNS, (3) añadir y eliminar direcciones de servidores DNS de la lista, (4) editar un servidor DNS de la lista, haciendo doble clic sobre él, (5) modificar el orden de los servidores DNS con los ascensores, y (6) añadir y eliminar un grupo de servidores DNS destinados a una red.

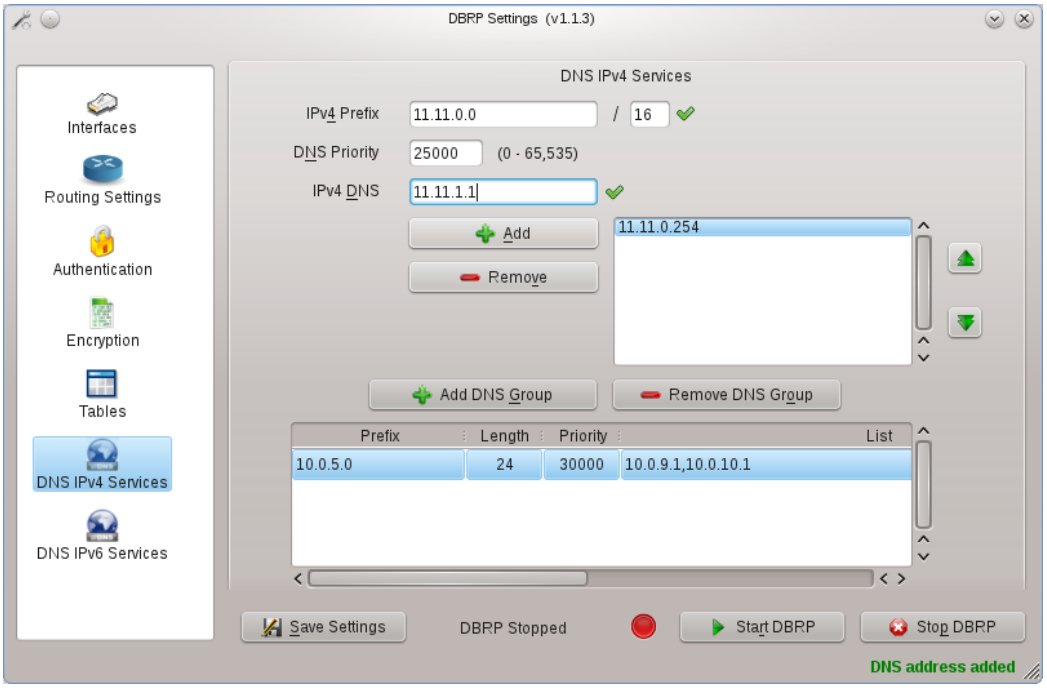

Figura 7.29: Módulo DNS IPv4 Services de DBRP Settings

- Módulo DNS IPv6 Services: permite configurar los servidores DNS destinados a determinada red y que serán transportados con DBRP (ver Figura 7.30). El usuario puede: (1) configurar el prefijo de la red destino para el cual están dirigidos los servidores DNS, validando el formato correcto, (2) establecer la prioridad de los servidores DNS, (3) añadir y eliminar direcciones de servidores DNS a la lista, (4) editar un servidor DNS de la lista, haciendo doble clic sobre él, (5) modificar el orden de los servidores DNS con los ascensores, y (6) añadir y eliminar un grupo de servidores DNS destinados a una red.

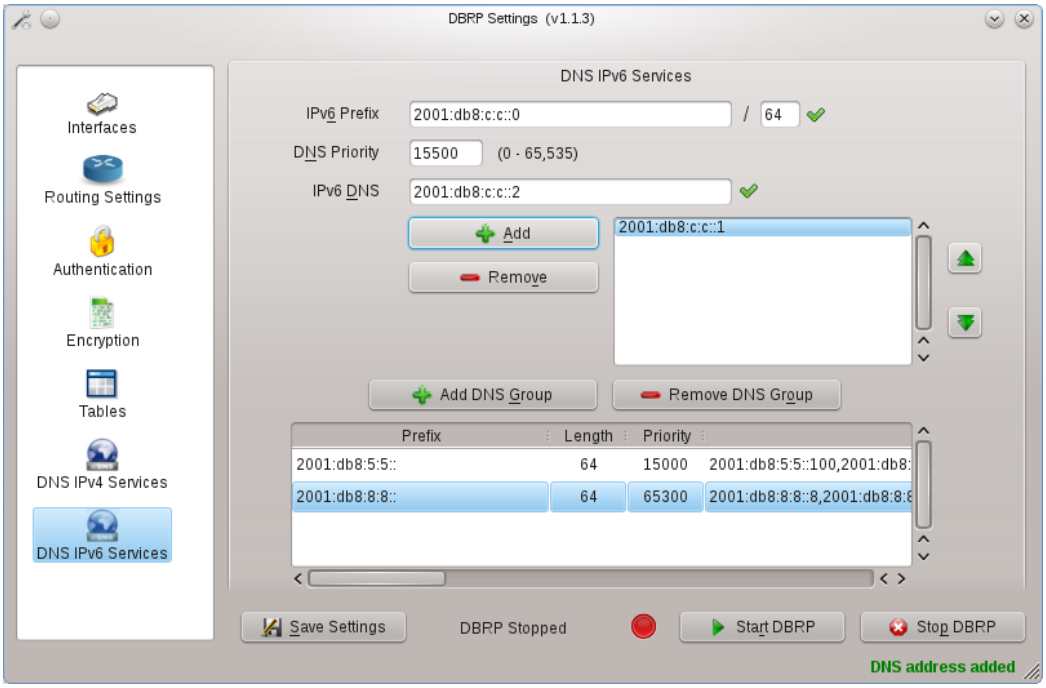

Figura 7.30: Módulo DNS IPv6 Services de DBRP Settings

Común a todos los módulos, en la parte inferior de la ventana se dispone de un panel de botones Save Settings, Start DBRP y Stop DBRP. Al presionar el botón Save Settings estando en cualquier módulo, se guarda la configuración actual al archivo de configuración, esta acción puede también hacerse mientras el protocolo está en ejecución. Los botones Start DBRP y Stop DBRP, inician y detienen el protocolo mostrando el estado de ejecución del mismo en todo momento con un indicador de estado. Adicionalmente se muestran mensajes en la barra de estado, referentes al módulo actual y a las acciones recientes de configuración.

- **Fase de Codificación:** cuando se inicia DBRP Settings, es leído el archivo de configuración y cargado en memoria, y al seleccionar un módulo se alimentan los elementos gráficos de la interfaz con esa información.

  Al seleccionar una interfaz de red en el módulo Interfaces, es posible visualizar y cambiar el estado de la misma, el comando usado para activar y desactivar se puede ver a continuación:

  ```
  # ip link set <interface_name> up|down
  ```

  Al seleccionar una interfaz de red se consultan al sistema operativo las direcciones IPv4 e IPv6 y sus máscaras, las cuales son mostradas en los campos correspondientes. Cuando alguna dirección IP o máscara es modificada, se valida en tiempo real si el formato es correcto y si no hay ningún solapamiento con alguna otra interfaz. Al presionar Save Settings se guardan los

valores de esos cambios en el archivo de configuración y se aplican en el sistema operativo siempre y cuando todas las validaciones hayan sido exitosas, de lo contrario se informa el error y los cambios no son aplicados.

Es posible agregar y eliminar interfaces virtuales, gracias al módulo dummy del kernel. Esta funcionalidad estará disponible sólo si este módulo está debidamente instalado en el sistema operativo. Estas interfaces pueden ser agregadas mientras el protocolo está en ejecución más no pueden ser eliminadas, debido al orden secuencial que mantiene el sistema operativo en la creación de estas interfaces.

Es obligatorio que todas las interfaces virtuales que se usen para DBRP sean creadas y eliminadas a través de DBRP Settings, esto se debe al nombre genérico que se le puede dar al módulo asociado a una interfaz virtual, si se crean interfaces virtuales fuera de DBRP Settings, no es posible conocer el nombre con el que fueron creadas.

Para distribuciones basadas en Debian, los comandos para crear y eliminar interfaces virtuales son mostrados a continuación:

```
# modprobe -o <name> dummy
```

```
# rmmod <name>
```

- **Fase de Pruebas:** en general todas las entradas de datos tales como rangos de campos numéricos, formatos de direcciones IP y longitud de claves fueron validadas para evitar configuraciones incorrectas.

  Es esencial que el archivo de configuración generado desde DBRP Settings contemple toda la configuración que el administrador estableció y que posea el formato adecuado para que pueda ser leído correctamente, esto fue verificado para cada uno de los módulos.

  Se verifica que la asignación de direcciones IP desde el módulo Interfaces se aplica correctamente en el sistema operativo y que al modificarlas no se acumulen múltiples direcciones en una interfaz.

  El uso de Qt Designer permitió depurar aspectos relacionados a la apariencia de la interfaz ya que permite editarla de manera gráfica, haciendo el proceso de diseño más eficiente.

7.2.6 Iteración 6: Interacción entre Proceso Demonio e Interfaz Gráfica

DBRP Settings tiene como función principal configurar el protocolo y visualizar el estado de sus tablas. Dado que el proceso demonio es el encargado de gestionar la

información de enrutamiento y las tablas, fue necesario sincronizar esta información. Al efectuar cambios en DBRP Settings y salvarlos, es necesario comunicar al proceso demonio que debe ser leído nuevamente el archivo de configuración para que estos cambios sean tomados en cuenta.

- **Fase de Diseño:** para lograr la sincronización de las tablas entre el proceso demonio y DBRP Settings fueron implementados hilos en cada una de las partes. Un primer hilo en el proceso demonio que mediante un proceso de serialización se encarga de encapsular la TE y enviarla a DBRP Settings a través de sockets de Unix para comunicación local. La TE es sincronizada periódicamente cada 1 segundo. Un segundo hilo en DBRP Settings se encarga de recibir la TE encapsulada, revertir el proceso de serialización y por último actualizar la información del módulo Tables.

  Una de las maneras en que el administrador puede manipular la ejecución del protocolo es a través de los botones Save Settings, Start DBRP y Stop DBRP de DBRP Settings, sin embargo es posible llevar a cabo estas acciones haciendo uso de un *script* diseñado para tal fin a través del intérprete de comandos.

- **Fase de Codificación:** cuando se presionan los botones Save Settings, Start DBRP y Stop DBRP es invocado el *script* dbrpd ubicado en directorio /etc/init.d del sistema de archivos de GNU/Linux. Este *script* se encarga de enviar señales específicas al proceso demonio dependiendo del parámetro con que sea llamado, como se puede ver a continuación:

```
# /etc/init.d/dbrpd  start|stop|reload
```

  En la ejecución del *script* son enviadas señales SIGUSR1 y SIGHUP con los valores descritos en la Tabla 7.1 para detener el proceso demonio y recargar el archivo de configuración respectivamente. Estas señales son enviadas a través del comando descrito a continuación:

```
# pkill  -SIGNAL  dbrpd
```

Signal	Value
SIGUSR1	10
SIGHUP	1

Tabla 7.1: Señales Enviadas al Proceso Demonio

  El proceso demonio recibe y atiende la señales a través de la función *signals().* Cuando se recibe una señal SIGUSR1 con valor 10, deben ser notificados todos los hilos *Client* y *Daemon* de cada interfaz de red, para que concluyan su ejecución adecuadamente. Cuando se recibe una señal SIGHUP se debe

determinar qué parámetros han cambiado en la configuración y así atender específicamente cada cambio.

- **Fase de Pruebas:** se efectuaron múltiples pruebas para verificar la sincronización entre la TE en memoria, la TE del sistema operativo y el módulo Tables de DBRP Settings. A pesar de ejecutar muchas veces el *script* dbrpd, se logró que éste sólo invocara una instancia del proceso demonio, para así garantizar la correcta ejecución del protocolo. Fue comprobada la recepción y atención de las señales del proceso demonio tanto desde DBRP Settings como desde el intérprete de comandos.

7.3 Fase de Pruebas y Depuración Final

Con el objetivo de validar el funcionamiento de DBRP, se plantean diversos escenarios en los que son evaluados un conjunto de aspectos. A continuación se muestran los escenarios más relevantes junto con una breve descripción de las pruebas realizadas.

7.3.1 Escenario 1

En este escenario se presenta una red en anillo de cinco PCs (ver Figura 7.31), donde cada uno cuenta con dos interfaces físicas conectadas a otros PCs y una interfaz virtual hacia una red *stub*. En este escenario se llevarán a cabo las actividades descritas en la Tabla 7.2.

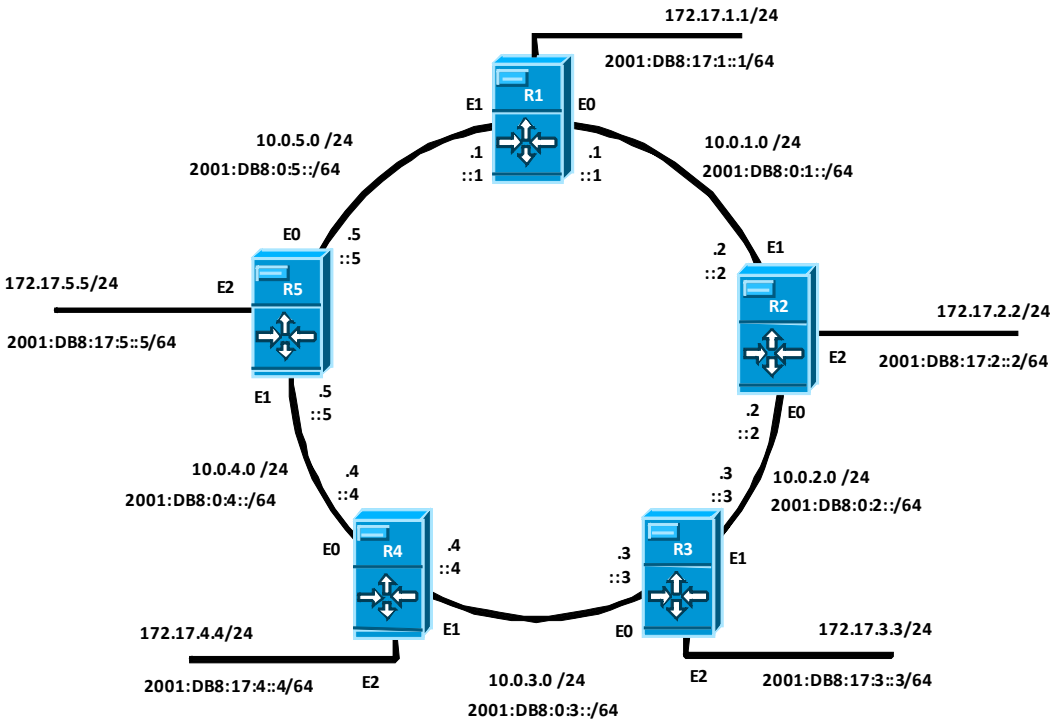

Figura 7.31: Topología en Anillo

Descripción de la prueba	Aspectos analizados	Errores encontrados
Configuración de direcciones IPv4/IPv6 en interfaces de todos los PCs a través de DBRP Settings.	Solapamiento de redes. Asignación de IPs.	No se verificaba solapamiento antes de salvar y ejecutar DBRP, fue corregido.
Inicio del protocolo en PC1, PC2 y PC3.	Tiempo de convergencia. Conectividad (correctitud de la TE).	Ninguno.
Inicio del protocolo en PC4 y PC5.	Tiempo de convergencia y conectividad. Agregación de nuevos vecinos.	Ninguno.
Modificación de direcciones IP y Máscaras en una interfaz virtual.	Tiempo de convergencia y conectividad. Propagación de cambios.	Ninguno.
Modificación de direcciones IP y Máscaras en una interfaz física.	Tiempo de convergencia y conectividad. Propagación de cambios.	Ninguno.
Desactivación/Activación de pila IPv4 o IPv6 en una interfaz virtual.	Tipo de mensaje enviado. Tiempo de convergencia y conectividad. Propagación de cambios.	Estos cambios no fueron contemplados dinámicamente, se incorporaron durante el desarrollo.
Desactivación/Activación de pila IPv4 o IPv6 en una interfaz física.	Tipo de mensaje enviado. Tiempo de convergencia y conectividad. Propagación de cambios.	Estos cambios no fueron contemplados dinámicamente, se incorporaron durante el desarrollo.
Desactivación/Activación de una interfaz física.	Tipo de mensaje enviado. Tiempo de convergencia y conectividad. Propagación de cambios.	Ninguno.
Detener el protocolo en un PC.	Tiempo de convergencia y conectividad. Evaluación del Shutdown.	En ocasiones no se actualizaba el status de ejecución de DBRP.

Tabla 7.2: Pruebas en Escenario 1

7.3.2 Escenario 2

Este escenario presenta una red con topología BMA (ver Figura 7.32), donde se reciben y envían mensajes de varios PCs por una misma interfaz física. Se evalúan las pruebas descritas en la Tabla 7.3.

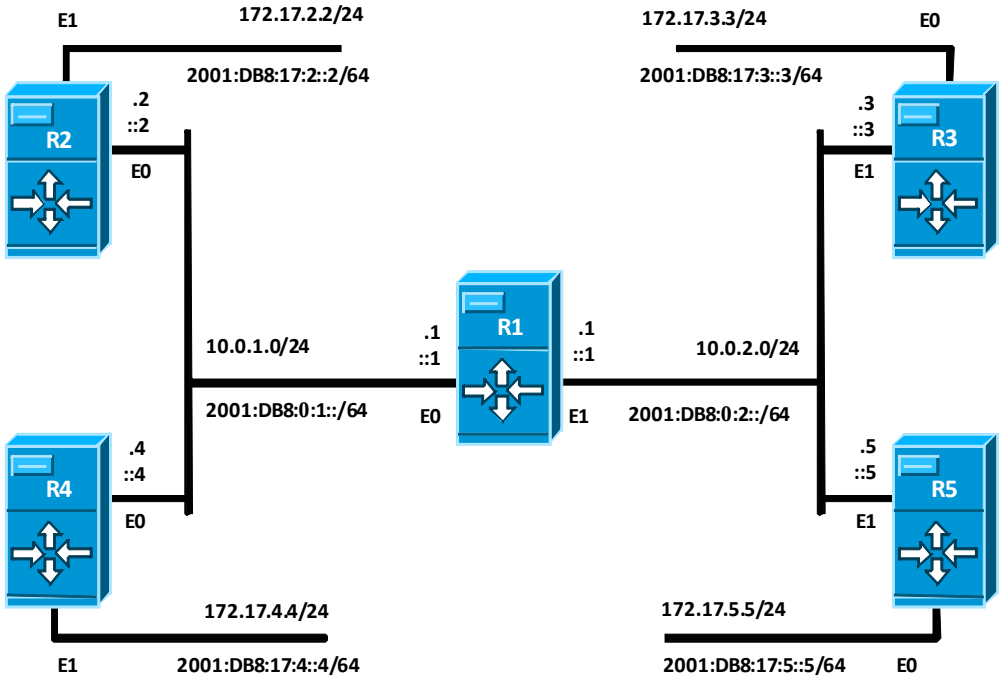

Figura 7.32: Topología BMA

Descripción de la prueba	Aspectos analizados	Errores encontrados
Configuración de direcciones IPv4/IPv6 en interfaces de todos los PCs a través de DBRP Settings. Configuración de autenticación y cifrado en PC1, PC2 y PC4 con claves y algoritmos distintos a PC3 y PC5.	Asignación de IPs. Correctitud de la TE.	En este escenario se implementó una tabla de vecinos para distinguir los mensajes recibidos por una misma interfaz proveniente de distintos routers.
Inicio del protocolo en PC1, PC2 y PC4.	Tiempo de convergencia y conectividad. Evaluación de parámetros de seguridad.	Ninguno.
Inicio del protocolo en PC3 y PC5.	Tiempo de convergencia y conectividad. Evaluación de parámetros de seguridad.	Ninguno.
Modificación de claves y algoritmos en PC3 y PC5 para igualarlos a PC1, PC2 y PC4.	Tiempo de convergencia y conectividad. Evaluación de parámetros de seguridad.	Ninguno.

Tabla 7.3: Pruebas en Escenario 2

7.3.3 Escenario 3

En este escenario se presenta una topología con ciclo (ver Figura 7.33), cada PC posee interfaces físicas para conectar a otros PCs e interfaces virtuales hacia una red *stub.* Se evalúan las pruebas descritas en la Tabla 7.4.

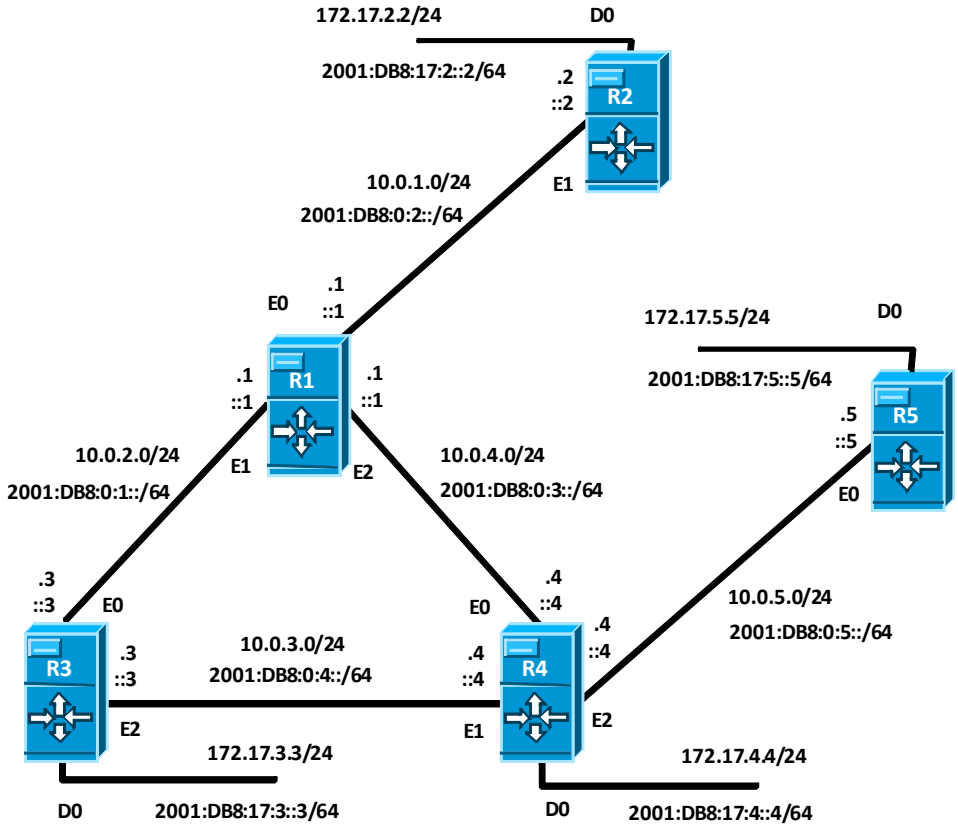

Figura 7.33: Topología con Ciclo

Descripción de la prueba	Aspectos analizados	Errores encontrados
Configuración de direcciones IPv4/IPv6 en interfaces de todos los PCs.	Correctitud de la TE.	Ninguno
Inicio del protocolo en todos los PCs.	Tiempo de convergencia y conectividad.	Ninguno
Se modifica el valor de la métrica en una interfaz física.	Propagación de cambios. Correctitud de la TE.	Al configurar una peor métrica en PC1, PC4 borraba el prefijo por un error en el mecanismo de *Holddown*. Fue corregido.
Es detenido abruptamente el protocolo en PC4.	Ciclos de enrutamiento, Split Horizon, Holddown. Tiempo de convergencia. Equivalencia de falla eléctrica, caída de enlace.	Ninguno
Es detenido adecuadamente el protocolo en PC4.	Tiempo de convergencia. Correctitud de la TE.	Ninguno

Tabla 7.4: Pruebas en Escenario 3

7.3.4 Evaluación de la Implementación de DBRP

Se consultó a un grupo de 9 estudiantes del área de redes mediante una encuesta descrita en el Capítulo Anexos, calificando la configuración de cada módulo del 1 al 5, siendo 5 el valor más alto. Las preguntas fueron:

1) Califique la distribución de los elementos
2) Califique la validación y recuperación ante errores
3) Califique la navegabilidad
4) Califique los mensajes de ayuda
5) Califique cuán intuitivo es este módulo

Los resultados obtenidos pueden ser visualizados a través de una gráfica que se muestra en la Figura 7.34. Se puede evidenciar el nivel de aceptación que tuvo la herramienta durante la configuración de los escenarios.

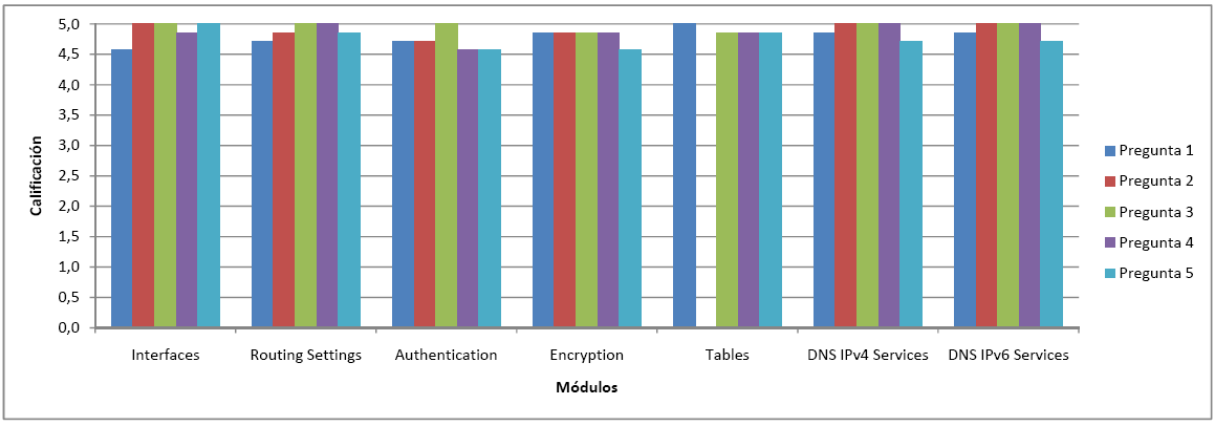

Figura 7.34: Resultados de Configuración por Módulo

También se consultó a este mismo grupo de estudiantes respecto a los escenarios de red en Anillo, BMA y Loop descritos en la Subsección 7.3, y se les preguntó:

6) Califique la sencillez de la configuración
7) Califique el comportamiento al hacer cambios dinámicos
8) Califique el tiempo de convergencia

En la Figura 7.35 se observan los resultados de esta evaluación.

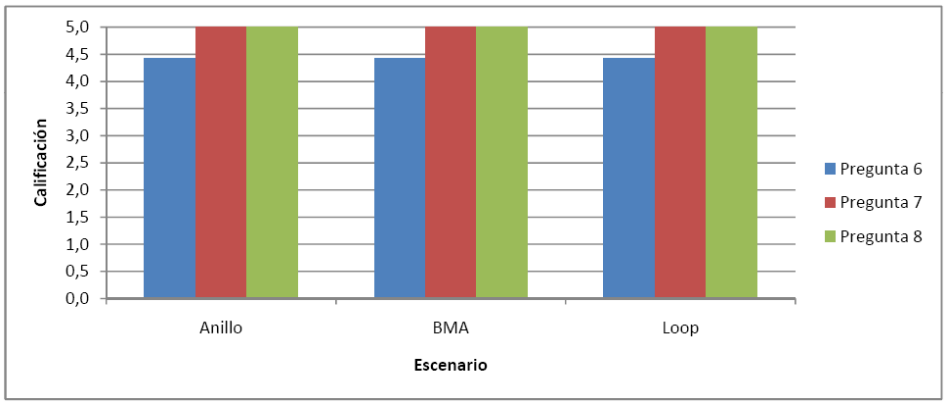

Figura 7.35: Resultados de Desempeño por Escenario

7.3.5 Complemento de Análisis para DBRP en Wireshark

Con el objetivo de hacer un análisis detallado del tráfico de red generado por DBRP, fue desarrollado un complemento o *plugin* para la disección de los PDUs definidos en la Subsección 5.1.1. La creación del complemento requiere la compilación del código fuente de Wireshark para GNU/Linux que está escrito en lenguaje C, para lo cual se eligió la versión más reciente para el momento (1.6.1). Su compilación es dependiente de la arquitectura. Este complemento (dbrp.so) por defecto debe estar ubicado en el siguiente directorio:

```
/usr/local/lib/wireshark/plugins/1.6.1/
```

Un mensaje DBRP es filtrado a través del campo *Ethertype* de Ethernet con valor 0x7777. Mediante un árbol es posible desplegar la cabecera y cada TLV, siendo posible detallar sus campos y valores con una breve descripción. Haciendo clic en un campo se sombrea la porción de datos que corresponde al mismo. Es posible también visualizar a nivel de bits aquellos campos que lo requieran.

Los pasos generales requeridos para el desarrollo de este complemento son:

- Inicialización de variables: Necesarias para almacenar y mostrar los datos extraídos del mensaje capturado. Deben ser declaradas e inicializadas globalmente.
- Función dissect: Encargada de realizar el análisis e interpretación detallada de los bytes en el mensaje.
- Lectura secuencial: Se lee el mensaje secuencialmente obteniendo datos de los tamaños requeridos según el campo a ser interpretado.
- Formato de representación: Se especifica en qué formato deben ser mostrados los campos.

Para información más precisa puede consultar el Capítulo 9 de la Guía de Desarrollo de Wireshark [14].

8. Especificaciones Técnicas de DBRP

El protocolo DBRP fue desarrollado para distribuciones GNU/Linux basadas en Debian para arquitecturas tanto x86 como x86_64, con entorno gráfico X Window. Su instalación se lleva a cabo a través de la utilidad dpkg como se muestra a continuación:

```
# dpkg -i dbrp-1.x_i386.deb
```

Es necesario cumplir previamente con las respectivas dependencias descritas en la Tabla 8.1.

Paquete	Versión	Descripción
g++	4.3.2-2+	Compilador C++ de GNU.
make	3.81-5	Utilidad “make” de GNU.
gdb	6.8-3+	GNU Debugger.
libqt4-dev	4.4.3-1+	Bibliotecas de desarrollo Qt 4.
libssl-dev	0.9.8g-15+	Bibliotecas de desarrollo SSL.

Tabla 8.1: Dependencias de Instalación del Paquete DBRP

Una vez que se ha configurado el protocolo a través de la herramienta gráfica DBRP Settings, encargada de que el archivo de configuración posea la sintaxis correcta, es posible ejecutar el proceso demonio sin necesidad de una interfaz gráfica, tal como se describe en la Subsección 7.2.6.

9. Conclusiones y Trabajos Futuros

Los primeros protocolos de enrutamiento de los años ochenta fueron de vector de distancia, lo cual no representaba un problema debido al reducido tamaño de las redes para ese entonces. A medida que las redes fueron abarcando más popularidad, su masificación se hizo común dado el apoyo que ofrecen para la distribución de la información. Entre los primeros protocolos de enrutamiento, se pueden mencionar RIP e IGRP. El primero tenía limitaciones con respecto a redes de gran tamaño y presentaba una lenta convergencia. El segundo introdujo mejoras para redes más grandes, sin embargo es de código propietario, al igual que su sucesor EIGRP. No sólo los protocolos de enrutamiento evolucionaban, sino también los protocolos de red en general, como fue el caso de IP (al pasar de IPv4 a IPv6), creando la necesidad de incorporar soporte para IPv6 en los protocolos de enrutamiento existentes. Este nuevo estándar, dio lugar a RIPng con soporte de IPv6 únicamente, y EIGRPv6 con soporte tanto para IPv6 como para IPv4.

Paralelamente otro tipo de protocolos de enrutamiento fueron ideados, este es el caso de IS-IS y OSPF, los cuales son de estado de enlace, que a diferencia de los de vector de distancia conocen la topología de la red. Sus implementaciones eran más completas pero a la vez más complejas. También, se desarrollaron versiones o modificaciones especiales de estos protocolos para dar soporte a IPv6, como OSPFv3 para IPv6 [4].

En este Trabajo Especial de Grado, se diseñó DBRP (*Delay Based Routing Protocol)*, como protocolo de enrutamiento de tipo vector de distancia, con soporte nativo para IPv4 e IPv6, ofreciendo mecanismos de seguridad, como autenticación y cifrado de la información de enrutamiento compartida entre los routers. DBRP utiliza el retardo como métrica y su diseño está basado en el uso de TLVs para proveer extensibilidad al protocolo. Su desarrollo es orientado a la plataforma GNU/Linux haciendo uso de sus aplicaciones y bibliotecas de desarrollo. Dada la estructura modular que ofrece el uso de TLVs, es posible añadir nuevas funcionalidades y mejoras al protocolo sin hacer mayores cambios en el diseño inicial del mismo.

DBRP cuenta además con una herramienta gráfica que permite configurar fácilmente los parámetros necesarios para el funcionamiento del protocolo, y fue desarrollado también un complemento o *plugin* para la disección de los PDUs de DBRP en Wireshark, filtrando a través del campo Ethertype de Ethernet. Mediante un árbol es posible desplegar la cabecera y cada TLV, siendo posible detallar sus campos y valores con una breve descripción, para un eficiente análisis y depuración del protocolo. Próximamente estarán disponibles para su descarga las versiones más recientes de la implementación de DBRP y el *plugin* de disección para Wireshark desde el sitio web de proyectos SourceForge[11].

[11] http://sourceforge.net/p/dbrp

Las pruebas llevadas a cabo con la implementación confirmaron la efectividad del diseño del protocolo. La implementación de DBRP realizada en este TEG fue probada por un grupo de estudiantes que dio un *feed-back* positivo sobre el protocolo, y su interfaz de configuración DBRP Settings.

Como trabajos futuros, se puede mencionar (1) la posibilidad de desarrollar un modelo de DBRP para un simulador de red como ns-3 [13] (*Network Simulator 3*) u OMNet++ [23], (2) el soporte de más servicios comunes, (3) el desarrollo de una implementación de DBRP para Windows, y (4) la extensión de DBRP a múltiples dominios. Esta última posibilidad es importante para la escalabilidad del protocolo y está contemplada desde la versión v1 del protocolo ya que la cabecera común de los PDUs tiene un campo llamado Domain Identifier que fue puesto con este fin.

Anexos

[Anexo A - Encuesta de Evaluación DBRPv1]

Encuesta de Evaluación DBRP v.1

Escenario de Configuración: Marque con "X" la casilla según su respuesta, siendo la puntuación más baja ① y la más alta ⑤.

Módulo Interfaces:

		①	②	③	④	⑤
1.	Califique la distribución de los elementos	1.					
2.	Califique la validación y recuperación ante errores	2.					
3.	Califique la navegabilidad	3.					
4.	Califique los mensajes de ayuda	4.					
5.	Califique cuán intuitivo es este módulo	5.					

6. ¿Qué considera hace falta en este módulo y por qué?

_____

Módulo Routing Settings:

		①	②	③	④	⑤
7.	Califique la distribución de los elementos	7.					
8.	Califique la validación y recuperación ante errores	8.					
9.	Califique la navegabilidad	9.					
10.	Califique los mensajes de ayuda	10.					
11.	Califique cuán intuitivo es este módulo	11.					

12. ¿Qué considera hace falta en este módulo y por qué?

_____

Módulo Authentication:

		①	②	③	④	⑤
13.	Califique la distribución de los elementos	13.					
14.	Califique la validación y recuperación ante errores	14.					
15.	Califique la navegabilidad	15.					
16.	Califique los mensajes de ayuda	16.					
17.	Califique cuán intuitivo es este módulo	17.					

18. ¿Qué considera hace falta en este módulo y por qué?

_____

Módulo Encryption:

		①	②	③	④	⑤
19.	Califique la distribución de los elementos	19.					
20.	Califique la validación y recuperación ante errores	20.					
21.	Califique la navegabilidad	21.					
22.	Califique los mensajes de ayuda	22.					
23.	Califique cuán intuitivo es este módulo	23.					

24. ¿Qué considera hace falta en este módulo y por qué?

_____

Módulo Tables:

		①	②	③	④	⑤
25.	Califique la distribución de los elementos	25.					
26.	Califique la navegabilidad	26.					
27.	Califique los mensajes de ayuda	27.					
28.	Califique cuán intuitivo es este módulo	28.					

29. ¿Qué considera hace falta en este módulo y por qué?

_____

Módulo DNS IPv4 Services:

		①	②	③	④	⑤
30.	Califique la distribución de los elementos	30.					
31.	Califique la validación y recuperación ante errores	31.					
32.	Califique la navegabilidad	32.					
33.	Califique los mensajes de ayuda	33.					
34.	Califique cuán intuitivo es este módulo	34.					

35. ¿Qué considera hace falta en este módulo y por qué?

_____

Módulo DNS IPv6 Services:

		①	②	③	④	⑤
36.	Califique la distribución de los elementos	36.					
37.	Califique la validación y recuperación ante errores	37.					
38.	Califique la navegabilidad	38.					
39.	Califique los mensajes de ayuda	39.					
40.	Califique cuán intuitivo es este módulo	40.					

41. ¿Qué considera hace falta en este módulo y por qué?

_____

Escenario de Red en Anillo: Marque con "X" la casilla según su respuesta, siendo la puntuación más baja ① y la más alta ⑤.

		①	②	③	④	⑤
42.	Califique la sencillez de la configuración	42.					
43.	Califique el comportamiento al hacer cambios dinámicos	43.					
44.	Califique el tiempo de convergencia	44.					

45. ¿Contemplaría algo adicional a este escenario?

_____

Escenario de Red BMA:

		①	②	③	④	⑤
46.	Califique la sencillez de la configuración	46.					
47.	Califique el comportamiento al hacer cambios dinámicos	47.					
48.	Califique el tiempo de convergencia	48.					

49. ¿Contemplaría algo adicional a este escenario?

_____

Escenario de Red con Ciclos:

		①	②	③	④	⑤
50.	Califique la sencillez de la configuración	50.					
51.	Califique el comportamiento al hacer cambios dinámicos	51.					
52.	Califique el tiempo de convergencia	52.					

53. ¿Contemplaría algo adicional a este escenario?

_____

Referencias Bibliográficas

[1] K. Beck and C. Andres. Extreme Programming Explained: Embrace Change. Addison-Wesley Professional. Segunda Edición. Noviembre, 2004.

[2] R. Callon. Use of OSI IS-IS for Routing in TCP/IP and Dual Environments. RFC 1195. IETF. Diciembre, 1990.

[3] A. Cockburn. Agile Software Development. Addison-Wesley Professional. Octubre, 2001.

[4] R. Coltun, D. Ferguson, and J. Moy. OSPF for IPv6. IETF. Diciembre, 1999.

[5] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. Introduction to Algorithms, Segunda Edición. MIT Press and McGraw-Hill. Diciembre, 2003.

[6] K. Downes, M. Ford, H. Liu, S. Spanier, and T. Stevenson. Internetworking Technologies Handbook. Second Edition. Cisco Press / Techmedia. Junio, 1999.

[7] J. Doyle and J. Carroll. CCIE Professional Development Routing TCP/IP, Volúmen I, Segunda Edición. Cisco Press. Octubre, 2005.

[8] J. Expósito, V. Trujillo, and E. Gamess. Easy-EIGRP: A Didactic Application for Teaching and Learning of the Enhanced Interior Gateway Routing Protocol. Sixth International Conference on Networking and Services. Cancun, Mexico. Marzo, 2010.

[9] H. Gredler and W. Goralski. The Complete IS-IS Routing Protocol. Springer. Diciembre, 2004.

[10] J. Guichard, W. ScottWainner, B. Weis, and D. McGrew. Internal Routing Protocol Support for Distributing Encryption Information. Cisco Technology, Inc. United States Patent N° US 7.620.975 B2. Noviembre, 2009.

[11] C. Hedrick. An Introduction to IGRP. Document ID 26825. Rutgers, The State University of New Jersey, Center for Computers and Information Services, Laboratory for Computer Science Research. Agosto, 1991.

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

[13] T. Henderson, S. Roy, S. Floyd, and G. Riley. ns-3 Project Goals. In Proceeding of the 2006 Workshop on ns-2: the IP Network Simulator. Pisa, Italia. Octubre, 2006.

[14] U. Lamping. Wireshark Developer's Guide: 35110 for Wireshark 1.4. Free Software Foundation. Marzo, 2010.

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

[16] G. Malkin. RIPng for IPv6. RFC 2080. IETF. Enero, 1997.

[17] D. Medhi and K. Ramasamy. Network Routing - Algorithms, Protocols, and Architectures. Morgan Kaufmann Publishers. Abril, 2007.

[18] D. Molkentin. The Book of Qt 4: The Art of Building Qt Applications. No Starch Press. Julio, 2007.

[19] J. Moy. OSPF Version 1. RFC 1131. IETF. Octubre, 1989.

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

[21] D. Oran. OSI IS-IS Intra-domain Routing Protocol. RFC 1142. IETF. Febrero, 1990.

[22] I. Pepelnjak. EIGRP Network Design Solutions. Cisco Press. Enero, 2000.

[23] M. Surhone, M. Tennoe, and S. Henssonow. OMNet++. Betascript Publishing. Noviembre, 2010.