UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN
LABORATORIO DE COMUNICACIÓN Y REDES

DISEÑO E IMPLEMENTACIÓN DE UN PROTOCOLO DE ESTADO DE ENLACE

Autores:

Jesús R. Urbáez M.
C.I. 14.411.258
E-mail: jesusurbaez@gmail.com

Arturo Palacios M.
C.I. 17.402.279
E-mail: arturo7200@gmail.com

Tutor: Prof. Eric Gamess

Octubre 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 Jesús Ramón Urbáez Malavé. C.I.: 14.411.258 y Arturo Palacios C.I.: 17.402.279, con el título "**Diseño e implementación de un protocolo de estado de enlace**", 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 _____, para que sus autores lo defiendan en forma pública, _____, 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 _____, dejándose también constancia de que actuó como Coordinador del Jurado el Profesor Tutor Eric Gamess.

_____
Prof. Eric Gamess
(Tutor)

_____ _____
Prof. _____ Prof. _____
(Jurado Principal) (Jurado Principal)

A Dios y a la Gran Virgencita de la Rosa Mística que siempre me han guiado, ayudándome en las buenas y en las malas.

Esto es para mis padres, Heraclio y Gladys, por soportarme, y esperar tanto tiempo por mí sin perder la confianza en que podía lograrlo.

A mi hermano Ramón y mi hermana Xiomary por ser aquellos que me brindaron una palabra de apoyo en el momento justo.

A Blanca Pérez, por soportar mis continuos cambios de humor con una valentía admirable y estar siempre a mi lado para cuando la necesitara 24/7.

A Jesús Expósito por ser una de las pocas manos amigas que estaba a la orden para ayudarnos en el desarrollo de este TEG.

A mis verdaderos amigos, que son pocos, por estar ahí siempre.

A la gran UCV por ser mi gran casa durante toda mi carrera.

A todos aquellos profesores que no se dejan llevar por las apariencias y las notas, sino que tratan de ver más allá en cada estudiante.

A mi tutor por su paciencia.

Jesús R. Urbáez M.

A lo largo de la vida uno se tropieza con muchos obstáculos que tiene que ir superando, así como realidades que se tienen que vivir y que no estamos a la espera de que nos puedan suceder. Pero de todas estas cosas uno aprende y se hace cada vez una persona más completa, no sólo en la vida académica sino también en la vida profesional y sentimental.

Más que un agradecimiento es una dedicatoria para las personas más importantes en mi vida que están ahora conmigo y las que por circunstancias del destino ya no están, pero siguen presentes en mi corazón. Por eso les dedico este TEG a las siguientes personas:

A Luis Palacios, mi Papa, ya que si no fuera por ti no estaría donde estoy y no fuera la persona que soy. Gracias por tu apoyo incondicional como padre, como amigo consecuente en las buenas y en las malas, cuidando siempre de mí. Por esas y muchas cosas más te dedico este logro.

A Laura Palacios, mi Hermana, por estar siempre pendiente y apoyándome con todo tu cariño y amor; peleando, disfrutando y viviendo a mi lado todos los días como un nuevo día lleno de metas por realizar.

A Zulay Moya de Palacios, mi Mama, por darme la vida, por darme una infancia feliz, queriéndome y amándome como madre. Aunque ya no te encuentres entre nosotros, sé que desde donde estés me sigues apoyando y cuidando.

A Ricardo Palacios, mi querido Hermano, la persona que siempre convivió conmigo, que juntos reímos, peleamos, lloramos, tomamos y tantas cosas más. Con el dolor de tu prematura partida, llevándote siempre en mi recuerdo y en mi corazón, te dedico y te doy el honor de este TEG. Compañero querido, siempre me harás falta, tu partida ha sido uno de los golpes más duros de mi vida, pero como siempre decíamos hermano: “Para adelante, que más se le hace”.

Agradezco a la UCV, mi casa de estudios, a mis Profesores de la Escuela de Computación y al tutor de este trabajo Profesor Eric Gamess por contribuir y formar parte de mis logros y haber hecho posible la realización que este TEG.

Un agradecimiento afectuoso y fraternal a mis compañeros de estudio. Amigos consecuentes en toda esta ruta de esfuerzos y alegrías que significo los años de estudio juntos.

Arturo Palacios Moya

Resumen

Título:
Diseño e Implementación de un Protocolo de Estado de Enlace.

Autores:
Jesús Urbáez Malavé
Arturo Palacios Moya

Tutor:
Profesor: Eric Gamess

El presente Trabajo Especial de Grado consiste en el desarrollo de una aplicación didáctica denominada Easy-OSPF para la enseñanza del protocolo OSPF (Open Shortest Path First). La aplicación provee un conjunto de herramientas que permiten estudiar los diferentes componentes y conceptos del protocolo antes mencionado, representándolos gráficamente para facilitar así el proceso de aprendizaje.

El trabajo está basado en el RFC 2328 de OSPF publicado en abril de 1998 por John Moy, el cual describe detalladamente el protocolo. Para su implementación se usó un proceso de desarrollo iterativo, respetando las fases de desarrollo de software: análisis, diseño, implementación, y pruebas.

Easy-OSPF está diseñado para trabajar en computadores con el sistema operativo Windows los cuales deben poseer condiciones mínimas para que la aplicación pueda tener un rendimiento óptimo al ejecutarse.

Easy-OSPF es una implementación del protocolo OSPF a nivel single-area donde se puede estudiar el proceso de sincronización de los routers, los parámetros principales del protocolo, la modificación de las tablas (enrutamiento, vecinos, y topológica), entre otras características propias de OSPF. Además le ofrece al usuario la posibilidad de configuración del protocolo, tanto al nivel de las interfaces como al nivel del router, a través de una interfaz gráfica de fácil manejo. Entre las herramientas que también se le proveen al usuario final, se encuentran los módulos de “Sniffer/Logger” (sniffer exclusivo de paquetes OSFP) y “Network Map” (graficación de la topología de la red), los cuales dan al usuario un conocimiento completo del estado del protocolo.

Palabras Claves: Easy-OSPF, OSPF, Protocolos de Enrutamiento, Herramienta Didáctica.

Tabla de contenido

Índice de Tablas

Índice de Figuras

Introducción

Con la evolución y el creciente uso de las tecnologías de comunicación, específicamente las redes, nació la necesidad de desarrollar procesos más eficientes para el intercambio de información en las mismas.

El conjunto de software desarrollado para manejar el intercambio de tráfico en las redes es denominado protocolos de enrutamiento. Básicamente son protocolos que recopilan información para establecer las rutas más eficientes para dicho intercambio. En la actualidad son muchos los protocolos de enrutamiento implementados, los cuales se adaptan a diferentes necesidades y tienen sus ventajas y desventajas con respecto al entorno en que se utilicen.

El protocolo OSPF (Open Shortest Path First) es quizás el protocolo de enrutamiento interior más usado hoy en día, en redes corporativas medianas y grandes. Es un protocolo muy interesante si se consideran las opciones y posibilidades de configuración que ofrece y que permite dar respuesta a escenarios o requerimientos diversos. Sin embargo, esa misma potencialidad exige del administrador de la red conocimientos y destrezas superiores a los que requiere la implementación de protocolos más simples.

OSPF es un protocolo de enrutamiento interior descrito en el RFC 2328. Es un estándar abierto, lo que hace que esté disponible en múltiples sistemas operativos: Windows 2003 Server, Unix, Cisco IOS, etc.

Dado que la mayoría de las implementaciones de software de OSPF están desarrolladas para trabajar bajo la plataforma Unix, nace el objetivo que motiva a realizar la presente investigación la cual comprende principalmente del desarrollo de una versión didáctica del protocolo de enrutamiento OSPF para usuarios del sistema operativo Windows.

Los capítulos planteados en el presente documento para el desarrollo de la propuesta son:

- Capítulo 1: Aquí se exponen los objetivos a lograr en el TEG y la problemática a resolver.
- Capítulo 2: En este capítulo se describe detalladamente la estructura y funcionamiento del protocolo de enrutamiento OSPF.
- Capítulo 3: Este capítulo muestra los trabajos relacionados al presente TEG ya existentes.
- Capítulo 4: Explica en forma general las estrategias a seguir en el desarrollo de la aplicación.
- Capítulo 5: Explica en detalle las diferentes etapas del proceso de implementación mediante la metodología utilizada.
- Capítulo 6: Este capítulo presenta los resultados de una encuesta realizada sobre un grupo de estudiantes que evaluaron la herramienta desarrollada.

- Capítulo 7: Presenta las conclusiones y recomendaciones obtenidas a partir del Trabajo Especial de Grado realizado.
- Capítulo 8: Presenta recomendaciones sobre la expansión del presente TEG y posibles trabajos a realizar en el futuro concernientes.

1 Definición del Problema

1.1 Planteamiento del Problema

Actualmente el protocolo de enrutamiento Open Shortest Path First (OSPF) es quizás el más usado en ambientes corporativos medianos y grandes. Es un protocolo muy interesante considerando las opciones y posibilidades de configuración que ofrece y que permite dar respuesta a escenarios o requerimientos diversos. Sin embargo, esa misma potencialidad exige del administrador de la red conocimientos y destrezas superiores a los que requiere la implementación de protocolos más simples. Por ser un protocolo tan complejo y robusto, su aprendizaje y compresión a fondo, pueden ser muy complejos.

Destacando la gran importancia de las redes de datos en ambientes de información y conocimiento, se hace indispensable facilitar su estudio, especialmente en aquellos protocolos que son muy utilizados, como por ejemplo OSPF, para que los estudiantes trabajen y experimenten con ellos; de allí la importancia de las herramientas didácticas que implementen protocolos y permitan su estudio bajo diferentes topologías de redes, facilitando el estudio personalizado por parte de los alumnos. Hoy en día no existen herramientas con enfoques didácticos que faciliten el proceso de aprendizaje del protocolo OSPF.

Es por ello que se estudiaron aplicaciones tanto software libre como propietarias que proveen la gestión de las tareas de OSPF y otros protocolos. Una vez analizadas se observó que las aplicaciones software libre no proporcionan la cantidad de herramientas suficientes y la mayoría de las mismas están implementadas en Unix. En cuanto a las aplicaciones propietarias existe una, pero posee limitaciones, y no está enfocada a la enseñanza del protocolo OSPF.

Debido a la problemática planteada se busca realizar una herramienta didáctica que cubra gran parte de los aspectos del protocolo de enrutamiento OSPF.

1.2 Justificación del Problema

En la actualidad la presencia de protocolos de enrutamiento tiene una importancia resaltante, ya que en la mayoría de sistemas y plataformas se hace uso de redes. OSPF goza de numerosas ventajas por lo que se ha convertido en un IGP (*Interior Gateway Protocol*) muy utilizado en nuestros días, pero las pocas implementaciones del protocolo para la plataforma Windows son propietarias y en consecuencia costosas. Además, ninguna de ellas está orientada al aprendizaje de OSPF, simplemente se limitan a desarrollar el protocolo con sus funcionalidades y se lo proveen al usuario para su despliegue. Por lo que la solución de este trabajo se enfoca en eliminar los altos costos y dificultades de aprendizaje de OSPF para esta plataforma.

La aplicación debe implementar OSPF de manera gráfica, y contemplar los lineamientos de usabilidad para así ofrecer al usuario final una herramienta realmente provechosa, que ponga a disposición distintas tareas relacionadas con el protocolo, permitiendo hacer un estudio en profundidad del mismo, así como también analizar cada uno de los procesos involucrados.

Por lo antes expuesto, surge la necesidad de implementar una herramienta didáctica exitosa que entre otras cosas tengan una buena facilidad de uso e instalación, versatilidad, interacción, con el fin de ser usada en el curso de redes, para aligerar el proceso de aprendizaje de OSPF.

1.3 Objetivos

1.3.1 Objetivo General

Diseñar e implementar una herramienta didáctica usable para la plataforma Windows que permita al usuario final obtener una mejor comprensión del funcionamiento del protocolo OSPF, así como proveer herramientas de configuración y administración del protocolo.

1.3.2 Objetivos Específicos

A continuación se enumeran los objetivos específicos del presente trabajo especial de grado:

- Estudiar a profundidad el protocolo de enrutamiento OSPF.
- Implementar del protocolo OSPF.
- Diseñar una interfaz gráfica que permita al usuario final configurar el protocolo.
- Implementar herramientas gráficas para ayudar a la comprensión del funcionamiento de OSPF.
- Desarrollar una herramienta didáctica que cumpla con las especificaciones del protocolo OSPF.
- Desarrollar un módulo para el manejo de logs de la herramienta para conservar registros de las actividades del protocolo.
- Implementar un módulo para observar el funcionamiento del protocolo en tiempo real.
- Realizar un sondeo de evaluación de la herramienta.
- Realizar pruebas en diferentes escenarios con el fin de depurar exhaustivamente la aplicación.

2 Open Shortest Path First (OSPF)

OSPF pertenece a la categoría de los protocolos de estado de enlace, los cuales se han erigido como una alternativa a los protocolos de vector de distancia utilizados en el pasado debido al incremento exponencial del tamaño de las redes y a los intempestivos cambios que pueden producirse en ellas. En la implementación de OSPF se pueden observar una serie de características propias del protocolo. Algunas de dichas características son: [3]

- El uso de áreas, lo cual disminuye el uso de recursos por parte de los routers.
- Soporte de VLSM y CIDR para el manejo eficiente de direcciones IPs.
- Balanceo de carga para rutas con costos iguales.
- La implementación de procesos de autenticación.

En este protocolo se debe observar que como parte fundamental del mismo se define una base de datos que describe la topología de la red. Cada router es responsable de detallar su base de datos a los demás routers del dominio de red al que está directamente conectado, a través de LSAs (*Link State Advertisements*). Estos LSAs son distribuidos por todo el dominio de red, que tomados de manera unificada, formarán la base de datos topológica.

Sin embargo, para intercambiar información entre los routers del dominio de red estos deben establecer una relación la cual los convertirá en “vecinos”. Esto se logra con el envío de paquetes pertenecientes al protocolo Hello. Al formarse esta relación los routers podrán comenzar el proceso de intercambio de información de sus respectivas bases de datos topológicas.

Una vez que un router tenga su base de datos topológica actualizada procede a calcular las rutas óptimas hacia cada router del dominio de red. Para determinar la ruta más corta entre un par de routers se toma en cuenta una métrica especialmente diseñada para OSPF, la cual es deducida en base al ancho de banda de los enlaces. Esta información, con respecto a la métrica, se procesa usando el algoritmo de Dijkstra para calcular las rutas más cortas.

Después de conocer los conceptos anteriormente expuestos se puede definir un comportamiento a muy alto nivel de OSPF en los siguientes puntos:

1. Todos los routers OSPF del dominio mandan paquetes Hello a través de sus interfaces a los demás routers OSPF. Si un par de routers se corresponden en ciertos parámetros estos se vuelven vecinos.
2. Entre vecinos OSPF se establece una relación de adyacencia la cual depende del tipo de red que estén conectados los routers en cuestión.
3. Cada router envía una serie de LSAs a todos sus vecinos, los cuales describen todos los enlaces de las interfaces del router emisor así como el estado de las mismas.

4. Cada router que recibe un LSA lo copia en su base de datos topológica y luego lo envía a todos sus vecinos.
5. Gracias al paso anterior cada router tiene una base de datos topológica idéntica.
6. Cuando la base de datos está completa cada router ejecuta el algoritmo SPF para calcular la ruta más corta hacia los demás routers conocidos del sistema autónomo. El grafo resultante se conoce como SPT (*Shortest Path Tree*).
7. Cada router construye su tabla de enrutamiento a partir del SPT.

Además de los componentes fundamentales mencionados anteriormente, para empezar a estudiar OSPF se tiene que tomar en cuenta algunas estrategias que fueron usadas en su implementación:

- OSPF enruta paquetes IP basados principalmente en la dirección IP encontrada en la cabecera de dicho protocolo. Por lo tanto los paquetes no son encapsulados por ningún otro protocolo mientras estén en el sistema autónomo.
- Cada router ejecuta de manera independiente un algoritmo para determinar el SPT en el que cada router se ve como raíz. Este árbol finalmente establece la ruta más corta desde ese router a cualquiera otro.
- OSPF permite dividir la interred en segmentos llamados áreas siendo su topología desconocida para el resto del sistema autónomo, lo cual reduce considerablemente el tráfico de enrutamiento.
- OSPF, como se mencionó antes, funciona bajo el protocolo IP con el campo *Protocol* puesto en 89. También usa la fragmentación/desfragmentación de IP ya que OSPF no provee estos mecanismos por sí mismo.

2.1 Aspectos Básicos

Algunos aspectos que se tienen que tomar en cuenta antes de estudiar a profundidad OSPF son todos aquellos conceptos externos que fueron aplicados al protocolo y que no fueron diseñados específicamente para él. Además se deben conocer algunas definiciones básicas del mismo antes de seguir.

2.1.1 Algoritmo de Dijkstra

Algoritmo desarrollado por E.W. Dijkstra que está diseñado para encontrar el camino más cortó entre dos nodos de un grafo. Generalmente los algoritmos de manipulación de grafos se dividen en dos tipos: métodos de fijación de etiquetas y métodos de corrección de etiquetas. Con fijación de etiquetas se refiere al establecimiento del costo o peso para ir de un nodo “v” a un nodo “w” los cuales son vecinos, es decir, se puede ir de “v” a “w” directamente sin pasar por otro nodo. Con corrección de etiquetas se habla de la posibilidad de cambiar cualquier etiqueta si hay una modificación en el grafo durante la aplicación del método. Este algoritmo se desarrolló orientado a la fijación de etiquetas siendo uno de los primeros para tal fin. Observando su implementación se puede explicar de la siguiente manera:

Se llamará “I” al nodo inicial y “F” al nodo final, dk (i) será la etiqueta del nodo “i” que indicará la distancia del nodo inicial al nodo “i” en la K-ésima iteración.

Paso 1: Se etiqueta permanentemente al nodo inicial con cero, d0 (I)=0, puesto que es la única distancia del nodo a el mismo. Para todos los demás nodos d0 (i)=infinito. La variable “p” recoge el último nodo etiquetado de forma permanente, en este primer paso p=I.

Paso 2: Para todos los nodos “j” conectados con el nodo “p”, es decir, existe el arco (p, j), se calcula su etiqueta temporal en la iteración K-ésima iteración:

$$dk(j) = \min \{dk-1(j), dk-1(p) + a(p, j)\}$$

Se puede observar que los nodos no conectados conservan la etiqueta de la iteración anterior:

$$dk(i) = dk-1(i)$$

Para todo nodo “i” no conectado con “p”.

Paso 3: Para todos los nodos “j” que no tienen etiqueta permanente, se busca el de menor etiqueta, que pasará a ser permanente y a ser el nodo “p”.

$$dk(p) = \min i = \{dk(j)\}$$

Se pasará de nuevo al paso 2 hasta que todos los nodos de la red tengan etiqueta permanente, que será la distancia del nodo inicial al nodo calculado.

2.1.2 Shortest Path Tree

Mientras no se configuren áreas en el sistema autónomo, cada router tiene una base de datos topológica idéntica y a su vez una representación de grafos única. Cada router genera su tabla de enrutamiento a partir de dicho grafo calculando un árbol de caminos más cortos viéndose a él mismo como raíz. Este árbol genera todas las rutas a cualquier destino en el sistema autónomo, sin embargo, sólo el próximo salto es utilizado en el proceso de transmisión.

En el proceso de construcción del árbol se tiene que tener en cuenta que sus ramas se van a dividir en 3 conjuntos:

- Tipo 1: Las ramas que ya fueron asignadas definitivamente al árbol en construcción.
- Tipo 2: Las ramas de las cuales se seleccionaran las próximas a formar parte del Tipo 1.
- Tipo 3: Las demás ramas.

Además los nodos se dividen en 2 conjuntos:

- Conjunto A: Los nodos conectados con las ramas de Tipo 1.
- Conjunto B: Los demás nodos.

Para comenzar con la construcción de árbol se escoge un nodo arbitrario colocándolo en el conjunto A. Seguidamente se colocan todas las ramas que estén conectadas a este nodo en el tipo 2. Se puede apreciar que el conjunto de ramas tipo 1 está vacío antes que nada. De aquí en adelante se realizarán los siguientes pasos:

1. La rama con la métrica más baja es inmediatamente colocada en el tipo 1, como consecuencia, un nodo pasará del conjunto B al conjunto A.
2. Estudiando las ramas que se conectan del nodo recién transferido al conjunto A con los nodos que aún están en el conjunto B. Si la rama bajo construcción es más larga que la de tipo 2 de dicho nodo se rechaza, si es más corta la rama del tipo 2 reemplaza a la que está en el tipo 1 en este momento.

Los pasos son repetidos hasta que el conjunto B de nodos y el conjunto de ramas tipo 2 estén vacios.

Adaptando el algoritmo a lo concerniente a OSPF se puede llegar a intuir que las ramas y sus costos son la analogía de los enlaces entre routers vecinos y sus correspondientes costos.

2.1.3 Equal-Cost Multipath

Una característica importante de OSPF es la capacidad de lidiar con dos o más rutas que tengan el mismo costo a un nodo, dado que esas rutas generarán un *next hop* distinto. Se puede llegar a pensar que dichas rutas aparecerán en la tabla de enrutamiento, lo cual sería imposible en la versión original del algoritmo de Dijkstra, por lo que se le han hecho pequeñas modificaciones para que acepte esta situación [17].

Estas modificaciones permiten que el tráfico hacia un determinado destino sea dividido en un número igual a tantas rutas con el mismo costo existan para llegar al destino en cuestión, sin embargo, esto puede llegar a ser contraproducente. Si las rutas por las cuales se va a dividir el tráfico constan de anchos de banda diferentes puede ocurrir que los paquetes lleguen con retrasos distintos afectando notablemente el rendimiento del sistema.

En conclusión, la opción de ECMP (*Equal-Cost MultiPath*) [18] es de gran ayuda en el balanceo de tráfico hacia un nodo, pero puede no serlo a la hora de la resolución de problemas en el dominio, por lo tanto, algunas implementaciones tratan de evitar el uso de ECMP tanto como puedan.

2.1.4 Métricas

El costo que OSPF le da a cada enlace está relacionado íntegramente al ancho de banda del mismo. OSPF divide 10^8 entre el ancho de banda correspondiente al enlace (siendo el resultado redondeado hacia abajo), como se puede ver en la Tabla 2.1.

Tecnología	Métrica
FastEthernet	1
Ethernet	10
64 Kbps	1562
128 Kbps	781
256 Kbps	390
T1	64
E1	48

Tabla 2.1: Tabla de Métricas

Esta métrica está almacenada sobre 16 bits, es decir, tiene un rango que va de 0 a 65.535.

2.2 Paquetes OSPF

Los paquetes OSPF consisten en múltiples encapsulaciones como lo muestra la Figura 2.1. El encapsulado más exterior es la cabecera IP de la cual OSPF aprovecha muchas de sus características, algunas de ellas son:

- OSPF utiliza los mecanismos de fragmentación/desfragmentación del protocolo IP puesto que no cuenta con dichos mecanismos por sí mismo.
- Puesto que los paquetes viajan no más de un salto, se suele colocar el campo TTL (*Time to Live*) en 1 para evitar reenvíos no necesarios.
- En la dirección IP de destino se coloca la dirección IP del vecino o de los grupos multicast *AllOSPFRouters* (224.0.0.5) o *ALLDRouters* (224.0.0.6) definidos por OSPF.

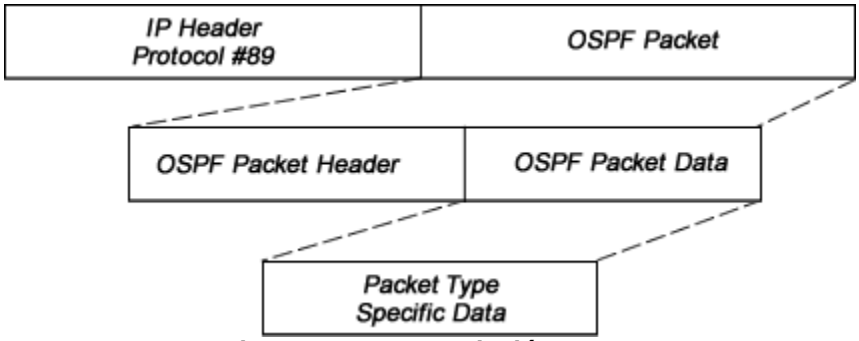

Figura 2.1: Encapsulación OSPF

Todos los paquetes OSPF tienen una cabecera común de 24 bytes que es descrita en la Figura 2.2 (Tomada de [5]).

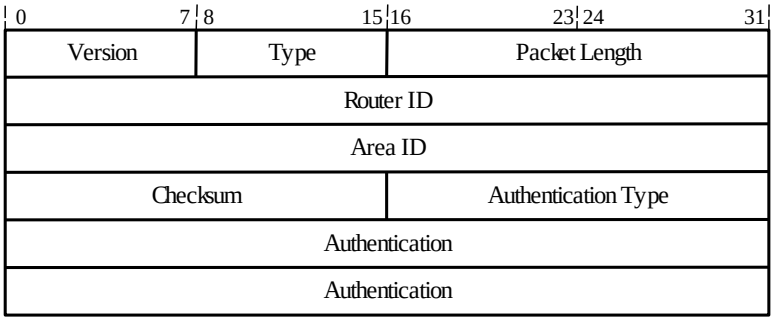

Figura 2.2: Cabecera Común OSPF

- Version: Versión del protocolo OSPF.
- Type: Tipo de paquete OSPF, existen 5 tipos brevemente mencionados en la Tabla 2.2.

Tipo	Descripción
1	Hello
2	Database Description
3	Link State Request
4	Link State Update

Tabla 2.2: Tipos de Paquetes OSPF

- Packet Lenght: Especifica la longitud del paquete OSPF en bytes.
- Router ID: Indica el ID del router que origina el paquete más no la interfaz desde la que fue enviado.
- Area ID: Especifica el ID del área donde se originó el paquete OSPF. El valor 0.0.0.0 es reservado para el backbone.
- Checksum: Checksum del paquete OSPF.
- Authentication Type y Authentication: Trabajan juntos para proveer la autenticación de los paquetes OSPF.

En cuanto al campo Authentication se puede ver en la Tabla 2.3 sus posibles valores.

Authentication Type	Significado	Campo Authentication
0	No hay Autenticación.	Puede ser cualquier cosa.
1	Autenticación simple enviada en texto plano.	Password de 8 bytes.
2	Autenticación usando MD5.	8 bytes divididos como se muestra en la Figura 2.3

Tabla 2.3: Valor del Campo Authentication Type

En la Figura 2.3 se puede apreciar el campo Authentication si el Authentication Type es igual a 2.

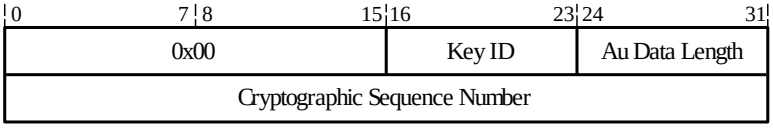

Figura 2.3: Campo Authentication

2.2.1 Protocolo Hello

Básicamente los propósitos del protocolo son:

- Ser el medio por el cual los vecinos son descubiertos.
- Negociar los parámetros por los cuales 2 routers se pueden convertir en vecinos.
- Indicar a los routers vecinos que el router que envía los paquetes Hello sigue activo.
- Asegurar la comunicación bidireccional entre los routers vecinos.
- Encargarse de elegir los DRs (*Designated Routers*) y los BDRs (*Backup Designated Routers*) en redes broadcast y NBMA (*Non Broadcast Multi Access*).

El protocolo Hello trabaja de manera distinta en redes broadcast, NBMA y punto-multipunto. En redes broadcast cada router envía paquetes Hello periódicamente para descubrir los routers vecinos de manera dinámica. En estos paquetes se envía el Router ID del DR así como los Router ID de aquellos routers que han enviado Hellos recientemente. En redes NMBA se necesita cierta configuración adicional puesto que cada router que puede convertirse en DR debe tener una lista de los routers de la red, entonces, cuando uno de ellos es designado DR envía paquetes Hello a los otros potenciales DR. En redes punto-multipunto un router envía paquetes Hello a todos sus vecinos. Estos vecinos se pueden descubrir dinámicamente a través de un protocolo como RARP (*Reverse Address Resolution Protocol*), o pueden ser directamente configurados. En la Figura 2.4 (tomada de [5]) se puede ver el paquete Hello.

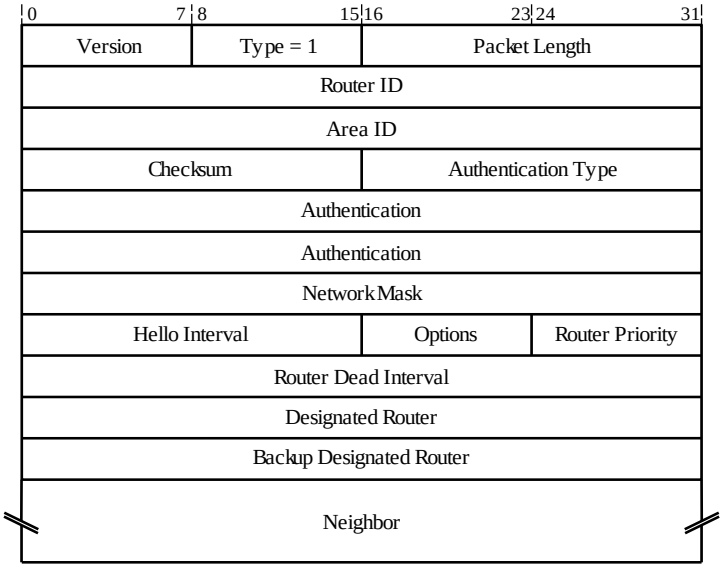

Figura 2.4: Paquete Hello

- Network Mask: Máscara de la interfaz donde se originó el paquete Hello.
- Hello Interval: Designa el tiempo en segundos entre 2 paquetes Hello. En redes punto-a-punto y redes broadcast el valor por defecto es 10 segundos mientras que en redes NBMA es 30 segundos.
- Options: En este tipo de paquetes sirve para establecer la compatibilidad entre routers OSPF. Sólo 5 bits han sido asignados (ver Figura 2.10).
- Router Priority: Indica la prioridad del router, es usado en el algoritmo de elección del DR y BDR.
- Router Dead Interval: Es la cantidad de tiempo que esperará un router para declarar a un vecino "muerto" puesto que no se ha recibido un paquete Hello de dicho vecino. El valor por defecto en redes punto-a-punto y broadcast es de 40 segundos mientras que en redes NBMA es de 120 segundos. Este valor debe ser aceptado por ambos en un principio para que no se rechace cuando un paquete Hello no concuerde dicho valor.
- Designated Router: Especifica la dirección IP de la interfaz del DR en la red. Si está es 0.0.0.0 significa que no hay DR.
- Backup Designated Router: Especifica la dirección IP de la interfaz del BDR. Si está en 0.0.0.0 significa que no ha sido escogido el BDR aún.
- Neighbor: Este campo es repetido por cada uno de los routers de los cuales se ha recibido un paquete Hello, es decir, contiene el Router ID de los vecinos.

2.2.2 Paquete Link State Request

Cuando un router detecta un cambio en la topología o que un LSA de su base de datos topológica ha caducado, este puede solicitar nuevos LSAs con el fin de mantener actualizada la base de datos. En la Figura 2.5 (tomada de [4]) se puede ver la disposición de los campos del LS Request.

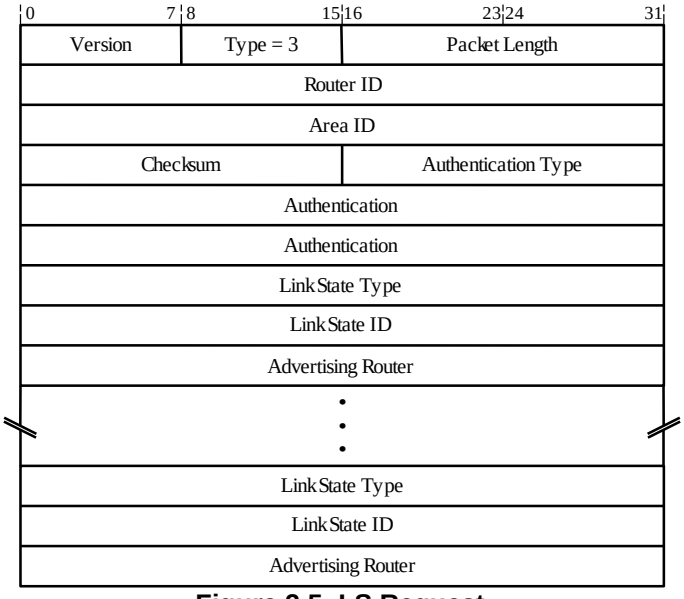

Figura 2.5: LS Request

- Link State Type: Identifica el tipo de LSA requerido.
- Link State ID: El contenido de este campo varía con el tipo de LSA como se puede ver en la Tabla 2.6.
- Advertising Router: El Router ID del router que originó el LSA.

2.2.3 Paquete Link State Update

Como se ha mencionado anteriormente, los demás paquetes OSPF (LS Request, LS ACK) no incluyen LSAs completos, sólo incluyen sus cabeceras. El paquete LS Update envía el contenido total de los LSAs requeridos por los vecinos que enviaron los respectivos LS Request. En la Figura 2.6 (tomada de [4]) se aprecia el paquete LS Update.

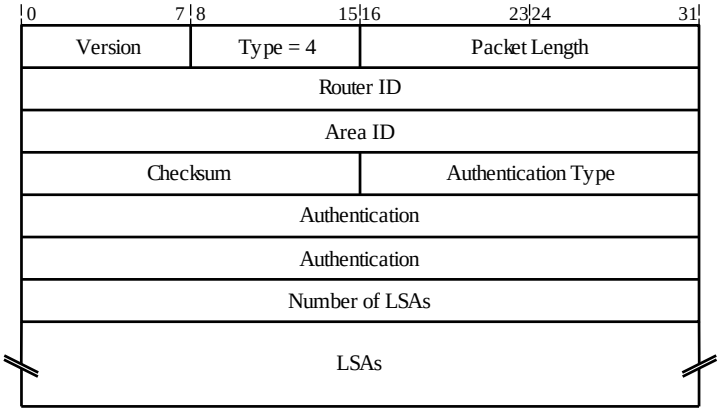

Figura 2.6: LS Update

- Number of LSAs: Indica el número de LSAs dentro del paquete.
- LSAs: En este campo se encuentran los LSAs de forma completa.

2.2.4 Paquete Link State ACK

La recepción de cada LSA debe ser confirmada mediante un LS ACK (*Link State Acknowledgement*), y ya que cada uno de dichos LSAs puede ser reconocido sólo por su cabecera, varios LSAs pueden ser reconocidos mediante un único LS ACK el cual contendrá una lista de dichas cabeceras como se puede observar en la Figura 2.7 (tomada de [4]). Básicamente estos paquetes son utilizados en el *flooding reliable.*

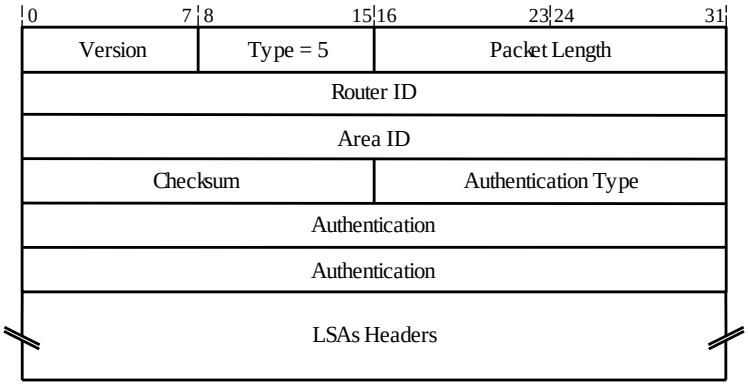

Figura 2.7: LS ACK

2.2.5 Paquete Database Description

Una vez que los routers OSPF hayan establecido sus adyacencias por medio de los paquetes Hello, los paquetes DBD (*Database Description*) son usados para que cada router pueda determinar si su respectiva base de datos topológica concuerda o no con la de los otros routers. OSPF usa para manejar el flujo de paquetes DBD una relación de maestro/esclavo entre los routers, la cual se determina al establecer las adyacencias, para así establecer un orden y sentido en la transmisión. Los campos del paquete DBD pueden apreciarse en la Figura 2.8.

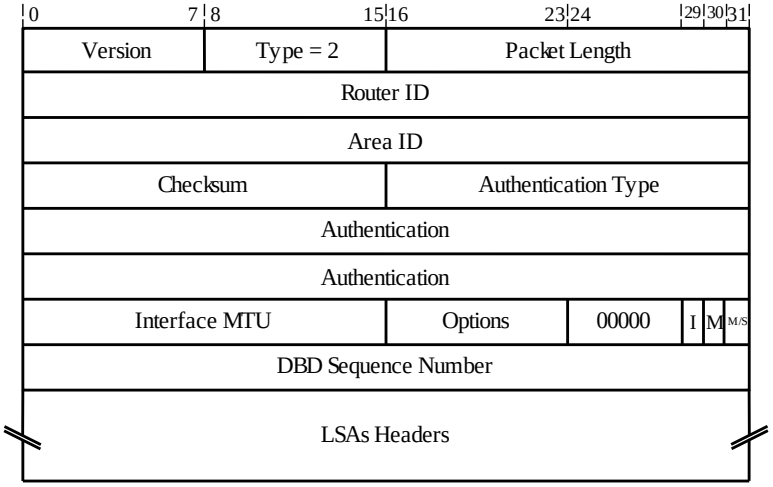

Figura 2.8: Paquete DBD

- Interface MTU (*Maximum Transmission Unit*): Indica la mayor longitud de un paquete que puede soportar la interfaz sin fragmentación.
- Options: Consiste de una serie de banderas (ver campo Options de la Sección 2.3).
- Bit I: El bit I (*Initial*) es puesto en 1, en el paquete inicial que empieza una sesión de sincronización de bases de datos; para los demás es colocado en 0.
- Bit M: El bit M (*More*) en 1 indica que no es el último paquete de la sesión de sincronización de bases de datos.
- Bit M/S: El bit MS (*Master/Slave*) es usado para indicar si el router que está enviando el paquete DBD es el maestro (si está en 1 el bit) o si es esclavo (si está en 0 el bit) en el proceso de intercambio de bases de datos topológicas.
- DBD Sequence Number: Es usado para incrementar el número de secuencia de los paquetes durante una sesión de sincronización de bases de datos. Este valor es establecido inicialmente por el maestro.
- LSA Headers: Contiene las cabeceras de los LSAs en la base de datos topológica del router origen. Puede listar algunos o todos ellos.

2.3 Link State Advertisements

Cada router en un sistema autónomo origina uno o más LSAs, los cuales tienen distintas funciones por separado pero en conjunto forman la base de datos topológica. Todos los LSAs de OSPF empiezan con una cabecera común de 20 bytes, la cual se muestra en la Figura 2.9 (tomada de [4]).

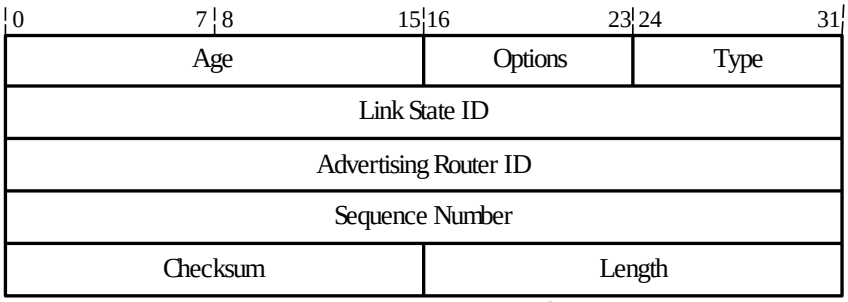

Figura 2.9: Cabecera Común LSA

- Age: Contiene el tiempo que ha vivido el LSA en segundos, el cual debe ser tratado como un entero de 16 bits sin signo. Se encuentra en 0 cuando es originado y es incrementado en cada salto que interviene en la inundación. Su rango se encuentra entre 0 y 30 minutos y si es sobrepasado se emitirá una nueva instancia del LSA incrementando su número de secuencia y colocando el valor del campo age en 0. En la Tabla 2.4 se pueden ver las acciones tomadas por un router dependiendo del valor del campo age.

Constante	Valor	Acción tomada por el Router OSPF
MinLSArrival	1 segundo	Tiempo mínimo que debe pasar antes que una nueva instancia de un LSA sea aceptada por el router.
MinLSInterval	5 segundos	Tiempo mínimo que debe pasar antes de se genere una actualización de un LSA en particular por parte del router.
CkeckAge	5 minutos	Cuando el campo age de un LSA contenido en la base de datos topológica muestra un múltiplo de esta constante el router procede a verificar de nuevo el checksum de dicho LSA.
MaxAgeDiff	15 minutos	Cuando 2 instancias de un LSA difieren más de 15 minutos se consideran instancias separadas, y será aceptada como la más reciente aquella con el campo LSA age más pequeño.
LSRefreshTime	30 minutos	El router debe refrescar cualquier LSA originado por el mismo que haya superado el rango de los 30 minutos.

MaxAge	1 hora	Cuando el LSA age llega a este valor el mismo es removido de la base de datos topológica.

Tabla 2.4: Parámetros de LSA Age

- Options: Este campo se encuentra presente en los paquetes Hello, en los paquetes DBD y en todos los LSAs contando con funciones diferentes para cada uno de ellos. Específicamente para los LSAs permite que los routers reenvíen o no el tráfico, a ciertos routers, con funcionalidades reducidas y la posibilidad de excluirlos del cálculo de las tablas de enrutamiento. Está compuesto de 8 bits de los cuales sólo 6 han sido asignados, lo que se puede ver en la Figura 2.10.

Figura 2.10: Campo Options

  - Bit DC: Toma el valor 1 cuando el router que origina el LSA soporta OSPF bajo demanda de circuitos.
  - Bit EA: Toma el valor 1 cuando el router que origina el LSA soporta LSAs del tipo 8 (ver Tabla 2.5).
  - Bit N/P: El bit N es usado sólo en los paquetes Hello. Cuando está puesto en 1 indica que el router soporta NSSA External LSAs. El bit P es usado por las cabeceras de los NSSA External LSAs y le indica al ABR (*Area Border Router*) que transforme este LSA en uno de tipo 5 (ver Tabla 2.5).
  - Bit MC: Toma el valor 1 cuando el router es capaz de retransmitir paquetes IP multicast.
  - Bit E: Toma el valor 1 cuando el router que origina el LSA es capaz de recibir AS External LSAs. También puede colocarse en 1 en todos los LSAs originados en el backbone y las NSSA. Este bit tomará el valor 0 en todos los LSAs originados dentro de un área Stub. Además es utilizado por los paquetes Hello para indicar la capacidad de una interfaz de enviar y recibir LSAs del tipo 5 (ver Tabla 2.5).
  - Bit T: Está puesto en 1 cuando el router que lo origina sea capaz de soportar TOS.

De la cabecera de los LSAs se utiliza la combinación de 3 campos (Type, Link State ID y Advertising Router) para identificar unívocamente el tipo.

- Type: Determina el formato y la función del LSA. Los tipos se describen en la Tabla 2.5.

Type	Descripción del LSA
1	**Router LSA:** Describen los estados de las interfaces de enrutamiento en el dominio.
2	**Network LSA:** Describe un segmento de red, por ejemplo, una red Broadcast o NBMA así como el conjunto de routers del dominio.
3	**Summary LSA:** Los LSA tipo 3 describen las rutas inter áreas.
4	**Summary LSA:** Los LSA tipo 4 describen las rutas a los ASBRs.
5	**AS External LSA:** Originados por los ASBRs y describen las rutas hacia destinos fuera del sistema autónomo.
6	**Group-membership LSA:** Indican la ubicación de los grupos multicast en MOSPF (Multicast OSPF).
7	**NSSA LSA:** Es usado en áreas NSSA para importar cierta cantidad de información de rutas externas.
8	**External attributes LSA:** Este tipo de LSA aún está siendo propuesto y sería usado para llevar información de rutas bajo el protocolo BGP a través de un dominio de enrutamiento OSPF.
9,10,11	**Opaque LSAs:** Tipo de LSA propuesto para intercambiar información de ciertas aplicaciones entre routers fuera del dominio OSPF y los routers que pertenecen a él.

Tabla 2.5: LSA Type

- Link State ID: Identifica la parte del dominio de enrutamiento que está siendo descrita en el LSA y depende intrínsecamente del campo Type. Esta relación de valores es observada en la Tabla 2.6.

Type	Link State ID.
1	El Router ID del router originario.
2	La dirección IP de la interfaz del DR.
3	La dirección IP de la red destino.
4	El Router ID de los routers AS boundary descritos.
5	La dirección IP de la red destino

Tabla 2.6: LSA Link State ID

- Advertising Router ID: Especifica el Router ID del router que originó el LSA.
- Sequence Number: En este campo se encuentra un entero de 32 bits con signo el cual, es usado para detectar LSAs viejos y duplicados. Cuando un router tiene dos instancias el LSA con el número de secuencia más alto será considerado como el más reciente.
- Checksum: Checksum del LSA completo, pero exceptuando al campo age ya que puede ser incrementado sin tener que actualizar el checksum. Es verificado en dos casos:
  1. Cuando es recibido en un paquete LS Update.
  2. Durante la actualización de la base de datos topológica.

- Lenght: Contiene el tamaño en bytes del LSA, contando tanto la cabecera del LSA como su contenido.

Puesto que se emiten diversas instancias de un LSA se debe velar por su correcta identificación, para lo cual se tienen los campos Sequence Number, Checksum y Age.

2.3.1 Router LSA

Cada router del sistema genera un Router LSA que contendrá todas sus interfaces y la información referente a ellas para compartirla con los routers vecinos. En la Figura 2.11 se pueden apreciar los campos del Router LSA.

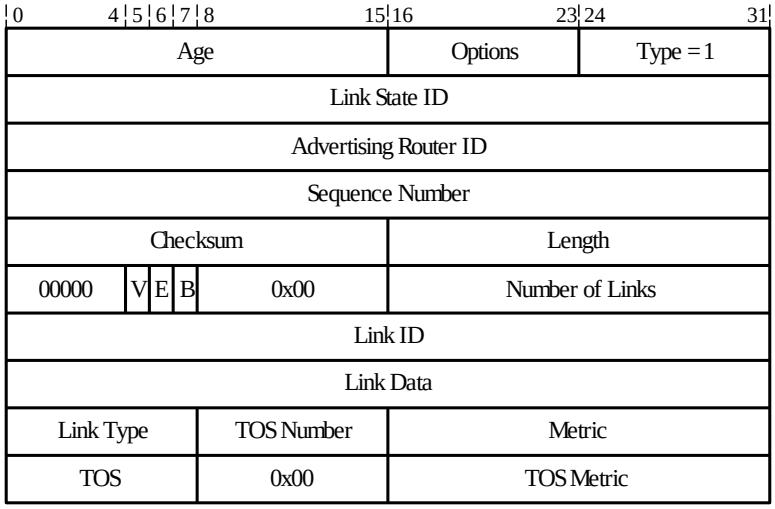

Figura 2.11: Router LSA

- Bit V: Indica si es un enlace virtual.
- Bit E: Indica si el router es un ASBR.
- Bit B: Indica si el router es un ABR.
- Number of Links: Indica el número total de enlaces que se están describiendo en el LSA.
- Metric: Es el costo del enlace. Tiene un rango de 1 a 65.535.
- TOS Number, TOS y TOS Metric: El número en el campo Number of TOS indica el número de Type Of Service incluido en el LSA, es decir, si TOS es 0 no hay TOS ni TOS Metric; si el valor es 2, TOS y TOS Metric se repiten 2 veces. Actualmente el valor cero es el único utilizado.
- Link Type: Determina el valor de los campos Link ID y Link Data como se puede ver en la Tabla 2.8 y la Tabla 2.9. Además se pueden observar los tipos de enlaces definidos para el campo Link Type en la Tabla 2.7.

Link Type	Tipo de Conexión	Comentario
1	Point-to-Point	Una conexión con otro router.
2	Transit Network	Carga con tránsito de datos.
3	Stub Network	Debe llevar paquetes de una fuente y/o destino local.
4	Virtual Link	Usado para enlazar áreas en el backbone.

Tabla 2.7: Tipos de Enlace

- Link ID: Depende del valor del campo Link Type, esto puede verse en la Tabla 2.8.

Link Type	Valor del campo Link ID
1	Router ID del router vecino (normalmente la dirección de loopback del router host).
2	Dirección IP de la interfaz del DR.
3	Dirección IP de la red.
4	Router ID del router vecino (normalmente la dirección de loopback del router host).

Tabla 2.8: Valores del Campo Link ID

- Link Data: También depende del campo Link Type como se ve en la Tabla 2.9.

Link Type	Valor del campo Link Data
1	Dirección IP de la interfaz del router que origina el LSA.
2	Dirección IP de la interfaz del router que origina el LSA.
3	Dirección IP de la red Stub o su máscara de subred.
4	Valor del MIB-II del enlace virtual.

Tabla 2.9: Valores del Campo Link Data

2.3.2 Network LSA

El network LSA es generado sólo por el DR y contiene el conjunto de routers conectados a una red broadcast o NBMA. El propósito de este tipo de LSA es asegurar que sólo un network LSA sea generado en la red para así disminuir el tráfico en la misma Figura 2.12 (tomada de [5]).

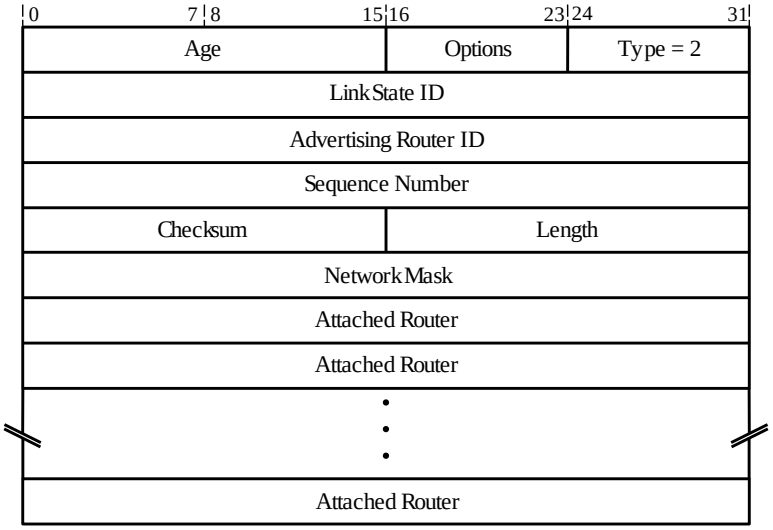

Figura 2.12: Network LSA

- Network Mask: Información estándar sobre máscaras.
- Attached Router: Contiene el Router ID de cada router que es full adyacente al DR de la red actual.

2.3.3 Summary LSA

Los Summary LSAs tipo 3 son generados por los ABRs para describir rutas inter áreas, es decir, describen redes que se encuentran en un dominio OSPF pero fuera de áreas OSPF en particular. Este tipo de LSA no es inundado dentro de la red que lo está anunciando. En la Figura 2.13 (tomada de [5]) se pueden ver los campos del Summary LSA tipo 3.

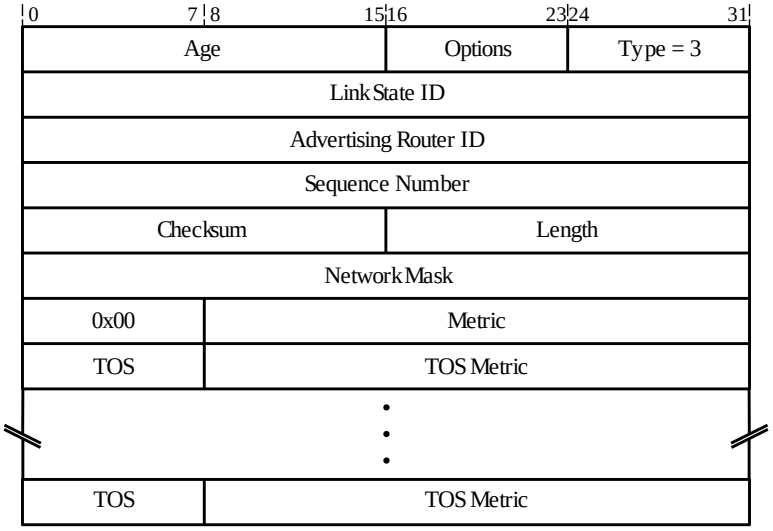

Figura 2.13: Summary LSA Tipo 3

- Link State ID: En este tipo de LSA contiene el identificador de la red o subred que está siendo anunciada.
- Metric: Es el costo a la ruta anunciada.

- TOS y TOS Metric: Campos opcionales.

Los Summary LSAs de tipo 4, al igual que los LSAs tipo 3, son originados por ABRs, pero en vez de anunciar una red anuncian como alcanzar a un ASBR. En la Figura 2.14 se pueden ver los campos que componen al Summary LSA tipo 4.

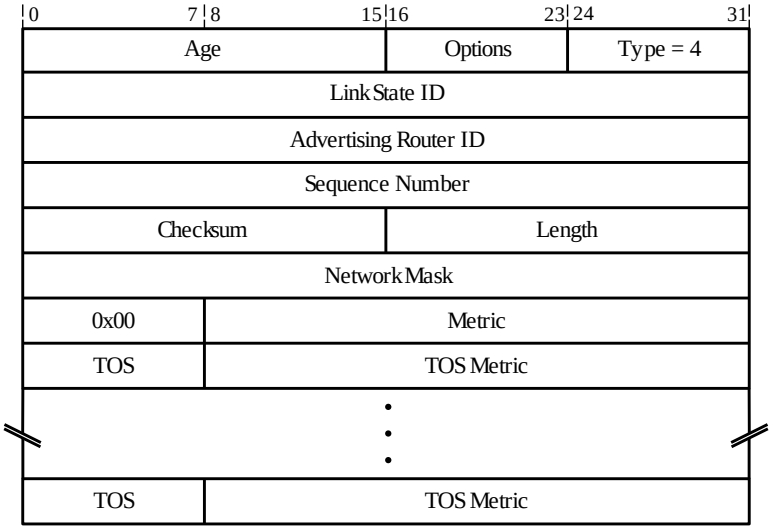

Figura 2.14: Summary LSA Tipo 4

- Link State ID: En este tipo de LSA contiene el Router ID del ASBR que está siendo anunciado.
- Network Mask: En este tipo de LSA no tiene aplicación.

En la Figura 2.15 se observa el funcionamiento del Summary LSA Tipo 4.

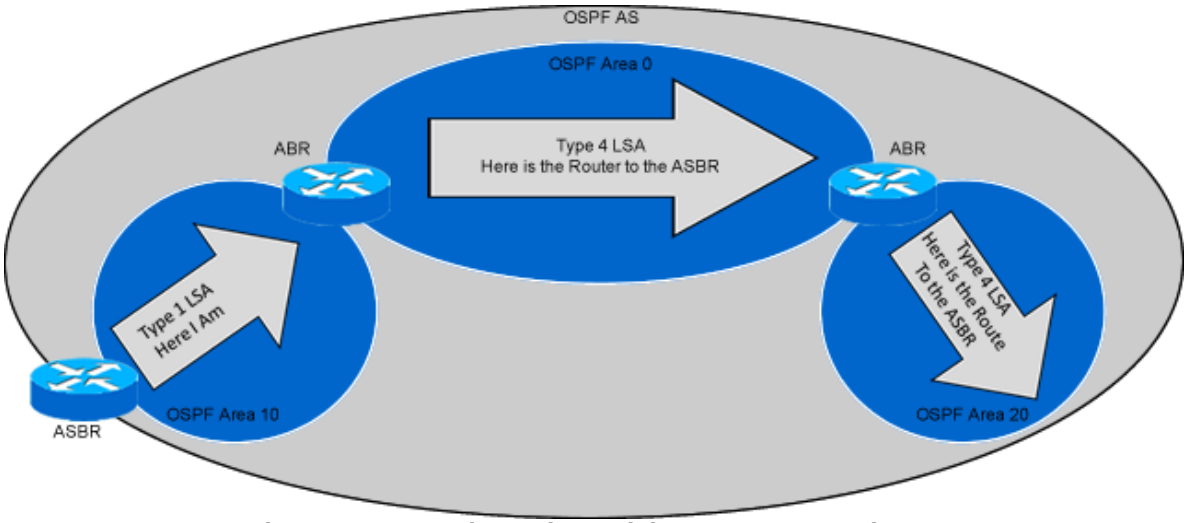

Figura 2.15: Funcionamiento del Summary LSA Tipo 4

2.3.4 AS External LSA

Estos LSAs son generados por ASBRs para describir rutas fuera del dominio OSPF. Este tipo de LSA es inundado a todas las áreas excepto a las de tipo Stub.

El Link State ID es la ruta que está siendo descrita y el Network Mask su correspondiente máscara. Las rutas por defecto utilizarán la dirección 0.0.0.0 en estos campos. El bit “E” es la métrica externa de la ruta, si el bit está en 1 la métrica será E2 si está en 0 sea E1. El tipo E1 incluye tanto el costo para llegar al ASBR como al destino en la ruta externa. El tipo E2 ignoran el costo para alcanzar el ASBR que originó el LSA. En la Figura 2.16 se puede apreciar la composición del AS External LSA.

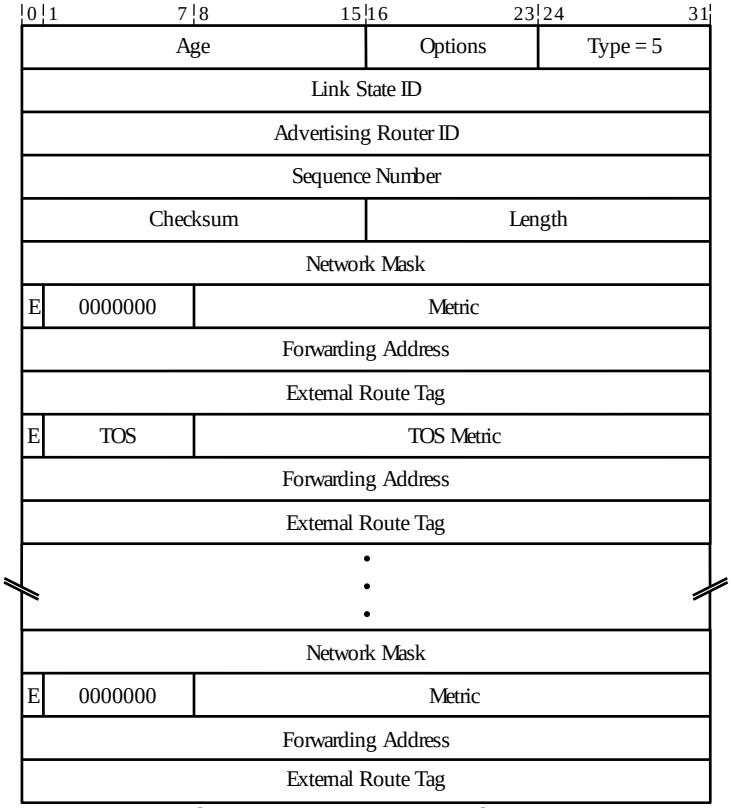

Figura 2.16: AS External LSA

Se puede notar que el formato de AS External LSA es muy similar al de los ABR Summary LSA a excepción de los siguientes campos:

- Forwarding Address: Contiene la IP por donde se envía el tráfico de datos. Si este campo en el router que lo envía lo coloca en 0.0.0.0 indica que para alcanzar la ruta externa especificada en el Link State ID se debe pasar por él.
- External Route Tag: Es usado para el intercambio de información entre routers fuera del dominio y los routers OSPF.

2.3.5 NSSA LSA

Estos LSAs de tipo 7 son generados por los ASBRs. Describen rutas externas al dominio OSPF y cuya descripción va a pasar por NSSA. Estos LSAs pueden ser sumarizados y convertirse en AS External LSAs. En la Figura 2.17 se pueden ver los campos del NSSA LSA. Los campos de los LSAs NSSA Area LSA coinciden con los LSA tipo 5.

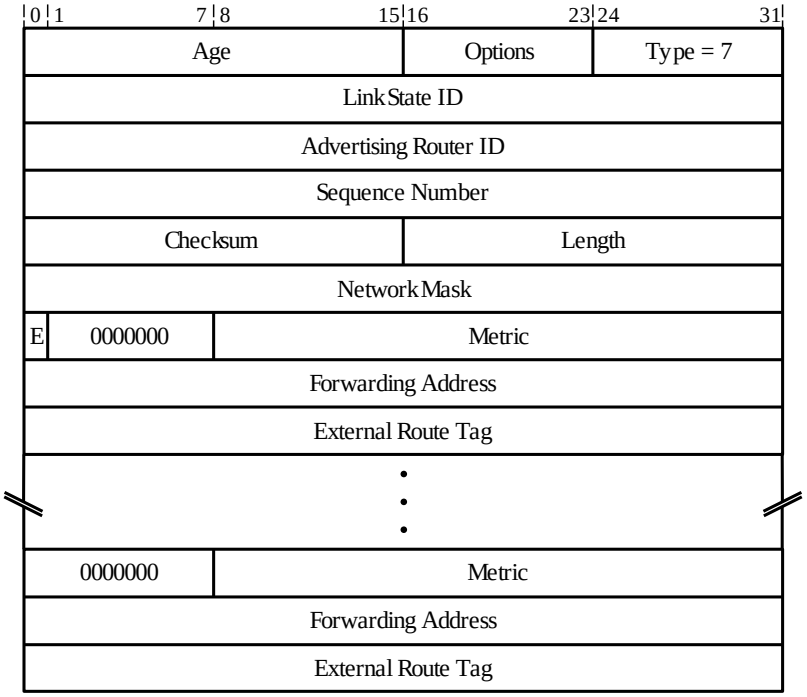

Figura 2.17: NSSA LSA

2.3.6 External Attributes LSA

Tipo de LSA propuesto como alternativa a los AS External LSA para anunciar rutas del protocolo BGP a través de un dominio OSPF. Los LSAs de tipo 8 no ha sido implementado aún.

2.3.7 Opaque LSAs

Tipos de LSA (9, 10, y 11) propuestos como estándar para llevar información de aplicaciones específicas. El campo Opaque Information puede ser usado por OSPF o indirectamente por otras aplicaciones para llevar información a través del dominio OSPF. En la Figura 2.18 se puede ver la estructura propuesta de estos LSAs.

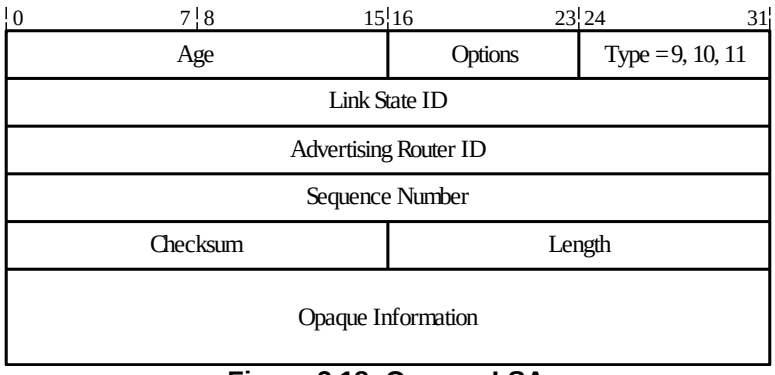

Figura 2.18: Opaque LSA

2.4 Base de Datos Topológica

Esta base de datos almacena los diferentes LSAs como un registro de los mismos, manteniendo cada router, dentro de un área determinada, una copia idéntica de dicha base de datos. De la información contenida en los LSAs los datos importantes para la creación de esta base de datos serán los concernientes a los Router IDs, los IDs de las redes y el costo de alcance a dichas redes ó routers vecinos. Estas bases de datos son intercambiadas inmediatamente después que los routers se descubren entre ellos y forman sus respectivas adyacencias.

De la Figura 2.19 se aprecia su base de datos topológica desde el punto de vista del router R0 (ver Figura 2.20) y desde el router R1 (ver Figura 2.21). Se puede ver en ambas bases de datos topológicas cada uno de los LSAs generados por los diferentes routers según la disposición de sus interfaces.

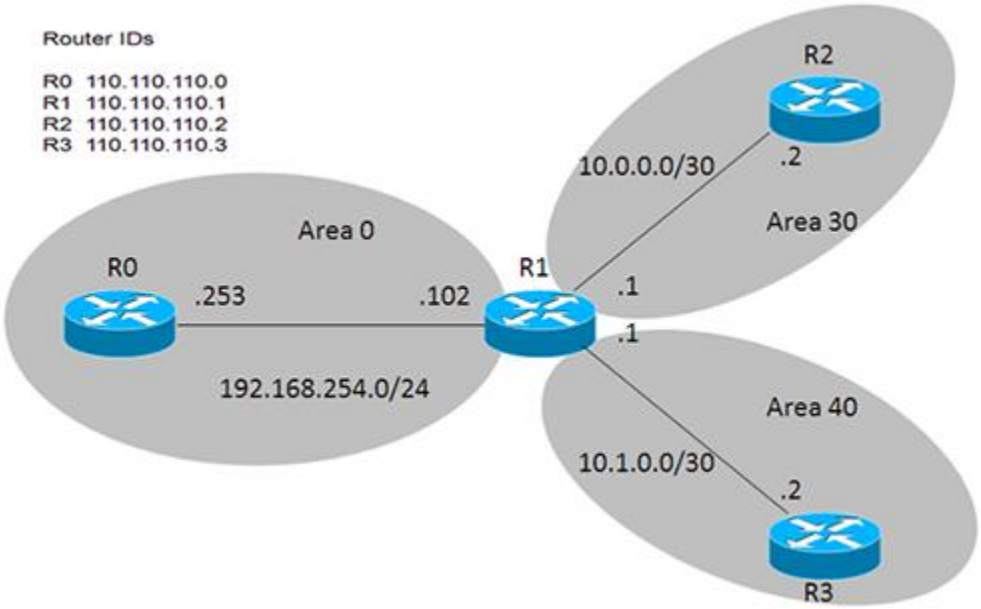

Figura 2.19: Topología

```
R0#show ip ospf database

       OSPF Router with ID (110.110.110.0) (Process ID 110)


                Router Link States (Area 0)

Link ID         ADV Router      Age         Seq#       Checksum Link count
110.110.110.0   110.110.110.0   365         0x80000002 0x90D5   1
110.110.110.1   110.110.110.1   364         0x80000002 0x39C1   1

                Net Link States (Area 0)

Link ID         ADV Router      Age         Seq#       Checksum
192.168.254.102 110.110.110.1   366         0x80000001 0x8403

                Summary Net Link States (Area 0)

Link ID         ADV Router      Age         Seq#       Checksum
10.0.0.0        110.110.110.1   346         0x80000003 0x9A4C
10.1.0.0        110.110.110.1   357         0x80000003 0x8E57
```

Figura 2.20: Base de Datos Topológica del Router R0

```
R1#show ip ospf database

       OSPF Router with ID (110.110.110.1) (Process ID 110)


                Router Link States (Area 0)

Link ID         ADV Router      Age         Seq#       Checksum Link count
110.110.110.0   110.110.110.0   15          0x80000002 0x90D5   1
110.110.110.1   110.110.110.1   16          0x80000002 0x39C1   1

                Net Link States (Area 0)

Link ID         ADV Router      Age         Seq#       Checksum
192.168.254.102 110.110.110.1   16          0x80000001 0x8403

                Summary Net Link States (Area 0)

Link ID         ADV Router      Age         Seq#       Checksum
10.0.0.0        110.110.110.1   2           0x80000001 0x9E4A
10.1.0.0        110.110.110.1   7           0x80000003 0x8E57

                Router Link States (Area 30)

Link ID         ADV Router      Age         Seq#       Checksum Link count
110.110.110.1   110.110.110.1   13          0x80000002 0xD1AF   1
110.110.110.2   110.110.110.2   21          0x80000002 0xCCB2   1

                Net Link States (Area 30)

Link ID         ADV Router      Age         Seq#       Checksum
10.0.0.2        110.110.110.2   21          0x80000001 0x94B5

                Summary Net Link States (Area 30)

Link ID         ADV Router      Age         Seq#       Checksum
10.1.0.0        110.110.110.1   17          0x80000003 0x8E57
192.168.254.0   110.110.110.1   18          0x80000003 0x88FC

                Router Link States (Area 40)

Link ID         ADV Router      Age         Seq#       Checksum Link count
110.110.110.1   110.110.110.1   25          0x80000002 0xE599   1
110.110.110.3   110.110.110.3   25          0x80000002 0xD0AA   1

                Net Link States (Area 40)

Link ID         ADV Router      Age         Seq#       Checksum
10.1.0.2        110.110.110.3   26          0x80000001 0x8CBA

                Summary Net Link States (Area 40)

Link ID         ADV Router      Age         Seq#       Checksum
10.0.0.0        110.110.110.1   11          0x80000003 0x9A4C
192.168.254.0   110.110.110.1   24          0x80000003 0x88FC
```

Figura 2.21: Base de Datos Topológica del Router R1

Uno de los aspectos más importantes del protocolo OSPF tiene que ver con la sincronización de las bases de datos topológicas de cada router. Si esta sincronización no se realiza de manera satisfactoria podrían presentarse errores de cálculo de rutas ó problemas de ciclos, entre otros. Este proceso se puede dividir en dos: primero, cuando dos routers vecinos empiezan a comunicarse deben sincronizar sus bases de datos topológicas antes de usar el medio para tráfico común; segundo, una continua sincronización de dichas bases de datos topológicas cuando se introducen nuevos LSAs, cambios en la topología, etc. En este tipo de sincronización se utiliza un mecanismo llamado *flooding reliable.*

2.4.1 Sincronización Inicial de la Base de Datos

Lo más común al pensar en la sincronización de la base de datos sería un intercambio completo de la misma entre todos los routers del sistema autónomo, lo que llevaría a un mal uso del ancho de banda, por lo cual este proceso se lleva a cabo sólo entre routers adyacentes enviando las cabeceras de los LSAs. Al proceso antes descrito se le conoce como “intercambio de bases de datos”.

Este intercambio se da una vez que el protocolo Hello determina que la comunicación es totalmente bidireccional, luego los routers que están intercambiando sus bases de datos tienen dos cosas por hacer: intercambiar todos los LSAs que tengan en sus respectivas bases de datos a través de una serie de paquetes e inundar el dominio con los LSAs actualizados que le lleguen producto de esta sincronización mediante el *flooding reliable*.

En efecto, después de ser recibidos todas las cabeceras LSAs por parte del router vecino, con el cual está intercambiando la base de datos topológica, el router está en conocimiento de cuales LSAs no tiene o cuales son más recientes. Luego le solicita al router vecino a través de paquetes LS Request los LSAs que necesita, a lo que el vecino responderá con paquetes LS Update que contendrán dichos LSAs de manera completa.

2.4.2 Flooding Reliable

Cuando un router quiere actualizar algún LSA originado por el mismo, (debido a cambios de estado, interfaces que se vuelvan inoperables etc.), debe inundar el dominio con un LS Update que puede o no contener otros LSAs.

Los routers vecinos que reciban el LS Update examinan su contenido y por cada LSA que contenga el paquete hará una nueva entrada en su base de datos topológica. Luego envía un LS ACK al router que envió el LS Update y procede a reempaquetar el LSA con una nueva cabecera LS Update luego de lo cual lo envía a través de todas sus interfaces sin incluir a la interfaz por la cual recibió el LS Update.

Además, hay que mencionar que esta inundación ocurre de manera distinta entre vecinos OSPF dependiendo de algunos factores que se nombrarán a continuación:

- Los LSAs del tipo 1 al 7 son inundados dentro de un área en particular.
- Los LSAs tipo 5 son inundados a través de todo el dominio OSPF excepto las áreas Stub o las NSSAs.
- Cuando existe un DR y un BDR, los DROthers sólo inundan hacia el DR y al BDR. Estos se encargarán de inundar los LSAs a los demás routers del dominio.

Cada router OSPF espera un reconocimiento para cada LS Update enviado, éste se puede enviar de manera explícita, enviando un LS ACK, o de manera implícita, el router envía el mismo LS Update al DR el cual lo retransmite a todo el dominio. El router que envió dicho LS Update al escucharlo llegar de vuelta lo toma como un ACK. Por cuestiones de confianza un router procederá a enviar periódicamente una retransmisión del LSA enviado al vecino hasta que le envíe un LS ACK listando los LSAs que le fueron enviados.

2.4.3 Manejo de Instancias de los LSAs

Cuando el router destino comprueba que los LSAs son más recientes que los que tiene reflejados en su base de datos topológica decide copiar estos nuevos valores para luego reenviarlos a sus vecinos.

Para llevar a cabo la comprobación el router ejecuta las siguientes acciones:

1. Compara los números de secuencia. El LSA con el número de secuencias mayor es tomado como el más reciente.
2. Si los números de secuencia son iguales, compara los checksums. El LSA con el checksum más alto será tomado como el más reciente.
3. Si los checksums son iguales, comparan los campos age. Si sólo uno de los LSAs tiene un age igual a *maxage* (3600 seg) es considerado el más reciente.
4. Si el paso anterior no es positivo determina si el valor de los campos Age de los LSAs difieren en más de 15 minutos (*MaxAgeDiff*), si esto es así el LSA con el age más bajo será el más reciente.
5. Si ninguno de estos casos aplica los LSAs son considerados los mismos.

2.5 Interfaces OSPF

Al configurar OSPF en un router sus interfaces pasan por una serie de estados antes de volverse completamente operacionales. Estos estados, los cuales se aprecian en la Figura 2.22, describen la relación del router con la red a la cual están conectados.

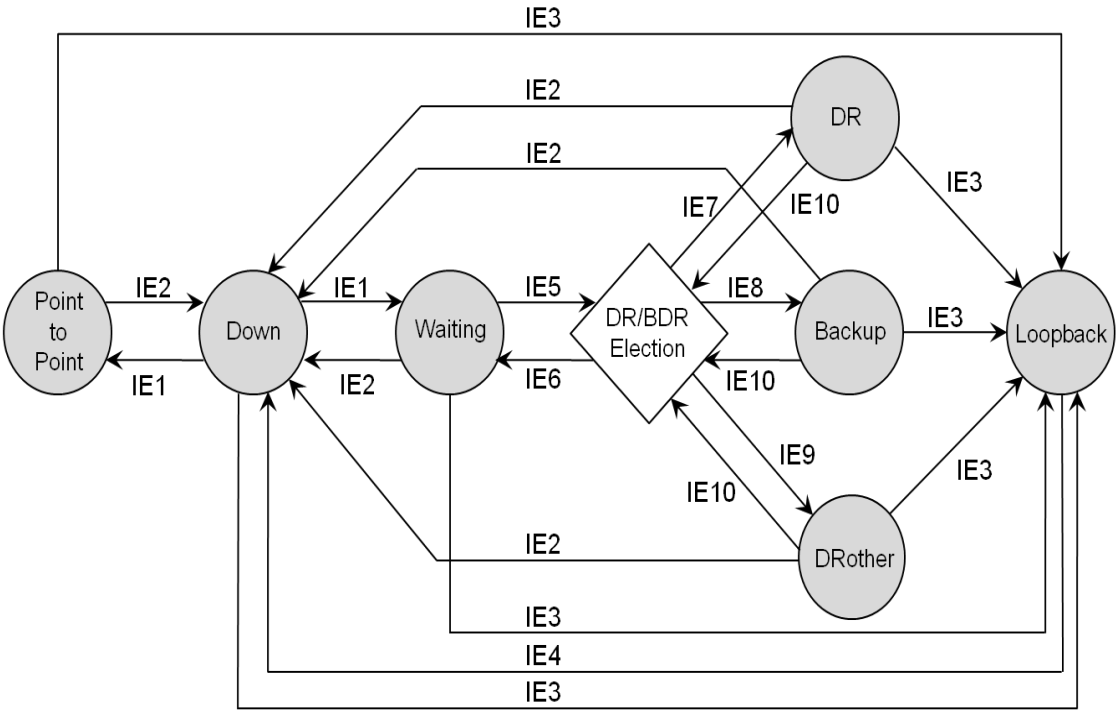

Figura 2.22: Máquina de Estado de las Interfaces OSPF

1. Down: Estado inicial de la interfaz en el que no se recibirá ni enviará tráfico, puesto que los protocolos de bajo nivel (ICMP) indican que no está operacional.
2. Loopback: En este estado la interfaz se vuelve de loopback. Esto puede realizarse mediante hardware o software. La interfaz sigue estando inutilizable para el tráfico regular.
3. Waiting: En este estado la interfaz comienza a recibir y enviar paquetes Hello. Además trata de determinar la identidad del DR y el BDR monitoreando los paquetes Hello que recibe. Este estado sólo es aplicable en redes broadcast o NBMA.
4. Point-to-Point: Este estado es válido sólo para routers en redes punto-a-punto, punto-a-multipunto o enlaces virtuales. Aquí la interfaz se vuelve operacional tratando de formar adyacencias con sus vecinos y enviando paquetes Hello cada *HelloInterval.*
5. DROther: En este estado la interfaz conoce la identidad del DR como la del BDR. Procede a formar adyacencias con estos.
6. Backup: En este estado la interfaz es elegida como BDR. Se establecen adyacencias con los demás routers del sistema autónomo.
7. DR: En este estado la interfaz es elegida como DR. Seguidamente se establecen adyacencias con todos los routers del sistema autónomo para luego originar un LSA Network que contendrá los enlaces a todos los routers del sistema.

Ahora se debe precisar los eventos que hacen que dichas interfaces cambien entre los estados definidos anteriormente, en la Tabla 2.10 se describen estos eventos como IEs (*Input Events*).

Input Event	Descripción
IE1	Los protocolos de bajo nivel indican que la interfaz se ha vuelto operacional.
IE2	Los protocolos de bajo nivel indican que la interfaz no es operacional.
IE3	Los manejadores de red de los protocolos de bajo nivel indican que la interfaz se ha vuelto de loopback
IE4	Los manejadores de red de los protocolos de bajo nivel indican que la interfaz ha dejado de ser de loopback
IE5	Es recibido un paquete Hello en el cual, el router que lo originó se enlista a él mismo como BDR ó DR. En el segundo caso indica que no hay BDR aún.
IE6	Expira el contador de espera.
IE7	El router es elegido como DR.
IE8	El router es elegido como BDR.
IE9	El router no es elegido como DR ni BDR.
IE10	Un cambio ha ocurrido en alguno de los vecinos. Puede ser alguno de los siguientes: • El establecimiento de comunicación full-dúplex con un vecino. • La pérdida de la comunicación full-dúplex con un vecino. • La recepción de un paquete Hello del DR o BDR en el cual no está listado como DR o BDR el router que lo originó. • La expiración del Router Dead Interval.

Tabla 2.10: Input Events

2.6 Formación de Adyacencias

Como se ha mencionado OSPF, establece adyacencias con los routers del sistema autónomo que cumplan ciertas condiciones. Sin embargo el proceso de establecimiento de adyacencias necesita de un conjunto de información para poder llevarse a cabo. Esta información se recaba de la estructura de datos que contiene cada router, la cual es aprendida de cada paquete Hello recibido.

Esta estructura de datos está constituida por los siguientes componentes:

- Neighbor ID: El Router ID del vecino.
- Neighbor IP Address: Dirección IP del vecino.
- Area ID: El ID que identifica el área.
- Interface: La interfaz del router conectada a la red donde el vecino se encuentra.
- Neighbor Priority: Es el *Router Priority* contenido en los paquetes Hello recibidos del correspondiente vecino.
- State: Estado funcional de la interfaz.

- PollInterval: Este valor sólo aplica a redes NBMA. Si el estado de la interfaz del vecino es *Down*, un paquete Hello será enviado a dicha interfaz cada PollInterval. Es un período un poco más largo que el *HelloInterval*.
- Neighbor Options: Las capacidades OSPF del vecino, discutidas en la Sección 2.3.
- Inactivity Timer: Contador que es puesto a cero cada vez que llega un paquete Hello. Si expira, la interfaz pasa al estado *Down.*
- Designated Router: Dirección IP de la interfaz del DR.
- Backup Designated Router: Dirección IP de la interfaz del BDR.
- Master/Slave: Esta relación se establece para determinar que router va a controlar el intercambio de las bases de datos topológicas.

2.6.1 Máquina de Estado para el Establecimiento de Adyacencias

Durante el proceso de formación de adyacencias, un router pasa por una serie de estados antes de declararse completamente adyacente a otro router:

1. Down: Ninguna información ha sido recibida por alguno de los routers.
2. Attempt: En redes NBMA indica que no se ha recibido información reciente por parte del vecino.
3. Init: El router ha detectado un paquete Hello proveniente del vecino, pero la comunicación bidireccional no se ha establecido.
4. Two Way: La comunicación bidireccional entre los routers se ha establecido. En el final de esta etapa, ya que tanto el DR como el BDR deben haber sido elegidos, los routers decidirán si formarán la adyacencia o no llevándose a cabo esta decisión tomando en cuenta si uno de los routers es el DR o BDR y si el enlace es PPP ó un enlace virtual.
5. Exstart: Los routers tratan de determinar el número de secuencia inicial que se va a usar en el intercambio de información. Además, uno de los routers se convertirá en maestro y el otro en esclavo, lo cual se decidirá por el Router ID más alto.
6. Exchange: Los routers intercambian la información de sus bases de datos.
7. Loading: Los routers finalizan el intercambio de información.
8. Full: La adyacencia se ha completado y los routers tendrán una misma base de datos topológica.

En la Figura 2.23 se puede observar la máquina de estado anteriormente descrita.

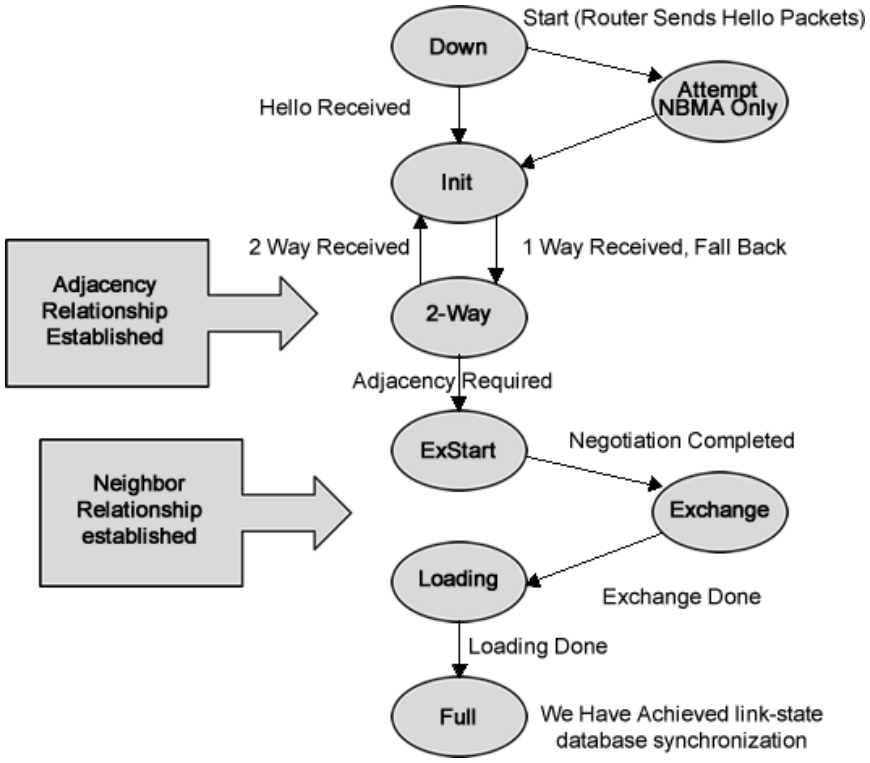

Figura 2.23: Máquina de Estados para el Establecimiento de Adyacencias

2.7 Áreas

OSPF permite agrupar de forma lógica grupos de routers para así limitar el tráfico entre ellos, ya que los cambios en un área en concreto no afectarían a las demás áreas, de esto se concluye que la topología de un área no es visible para un router que no pertenezca a ella. Cada área corre por separado el algoritmo básico de enrutamiento de estado de enlace, lo que significa que cada una de ellas posee su propia base de datos topológica. Se pueden establecer tres tipos de áreas especiales dentro del protocolo OSPF:

- Backbone: Llamada también área 0. Esta área especial tiene la peculiaridad de contener a todos los ABRs del sistema. El backbone es el responsable del enrutamiento inter áreas.
- Área Stub: Área donde la información sobre rutas externas, comunicadas a través de ASBRs, no es enviada. El ABR (que es único) de esta área es la ruta externa por defecto. En la Figura 2.24 se puede apreciar que el Área 1 y 2 son Áreas Stub. Dentro de estas áreas los routers deben llevar el bit E de paquete Hello puesto en 0 para formar sus respectivas adyacencias.
- Área Not-so-stubby: Tipo especial de área Stub que puede importar rutas externas de sistemas autónomos y enviarlas al backbone, pero no puede recibir rutas externas desde el backbone u otras áreas.

Aquellas áreas que no posean las características de alguna de las tres definidas anteriormente simplemente serán llamadas “áreas”, diferenciándolas por su Area ID.

Nota: Las rutas externas se definen como rutas que fueron inyectadas en el protocolo OSPF desde otro protocolo de enrutamiento.

2.7.1 Enrutamiento Inter Áreas

Cuando se tiene que enrutar un paquete entre dos áreas, de las cuales ninguna es el backbone, se procede a seccionar la ruta en tres sub rutas. Una ruta intra área desde el router del área fuente, otra sub ruta en el backbone y finalmente una sub ruta en el área destino. Cada router de borde de un área sumariza los costos, mediante Summary LSAs, hacia las redes externas. Después que el árbol SPF es calculado para el área en cuestión las rutas hacia todos los destinos inter áreas son calculados examinando la sumarización del router de borde dicha área.

2.8 Tipos de Routers

OSPF describe varios tipos de routers en su estructura jerárquica. Cada uno de estos routers tiene un rol especial dentro de dicha estructura además de una serie de características que los identifican.

2.8.1 Router Designado

Si se establecieran todas las adyacencias posibles en un dominio de tamaño considerable, se observaría un crecimiento desproporcionado de la base de datos topológica debido a la cantidad de LSAs que se producirían. Para reducir este problema se creó la figura de un “pseudonodo”, llamado DR (*Designated Router*) el cual se encargará de lidiar con el proceso de inundación de los LSAs que se produzcan en el sistema. El DR es designado por el protocolo Hello.

Cada red broadcast o NBMA tiene un DR, el cual tiene dos funciones principales:

- Se convierte en adyacente de todos los otros routers de la red.
- Origina un LSA Network el cual contiene a todos los routers que están incluidos en la red (hasta el mismo). El Link State ID de este LSA es la dirección IP de la interfaz del DR.
- Todos los LSAs del dominio salen de él.

2.8.2 Router Designado de Backup

Pueden ocurrir situaciones en la que se debe hacer una transición a un nuevo DR para esto se definió la figura de un BDR (*Backup Designated Router*). El BDR es también adyacente a todo los routers de la red.

Si no existiera el BDR, al fallar el correspondiente DR de la red tendría que comenzar un proceso de formación de nuevas adyacencias con el nuevo DR y por lo tanto, un proceso de sincronización de la base de datos de topológica que podría llevar mucho tiempo durante el cual la red no estaría disponible para su

uso. El BDR es también elegido por el protocolo Hello, por ende en cada paquete Hello hay un campo que especifica el BDR de la red.

2.8.3 Algoritmo de Elección del DR y BDR

Con este algoritmo, el cual es invocado por la máquina de estado de la interfaz, se designan al DR y al BDR de cada área. El proceso de elección del DR y el BDR se realiza utilizando los paquetes Hello:

1. Después de establecer una comunicación bidireccional con uno o más routers vecinos se examinan sus respectivos campos de prioridad para listar cuáles de ellos son elegibles como DR (se dice que un router es elegible cuando su prioridad es mayor a 0), luego todos los routers elegibles se declaran a sí mismos como DR (colocan la dirección IP de su respectiva interfaz en el campo Designated Router del paquete Hello).
2. De la lista de routers elegibles para DR se toma un subconjunto de ellos.
3. Si uno o más de los routers de ese subconjunto tomado anteriormente se declaró candidato a BDR (colocó su dirección IP de la interfaz en el campo BDR del paquete Hello), aquel que tenga la mayor prioridad será declarado BDR. En caso de empate aquel router que posea el mayor Router ID será designado BDR.
4. Si de la lista de routers elegibles para DR no hay ninguno que se declare candidato a BDR, aquel que tenga la mayor prioridad será declarado BDR. En caso de empate aquel router que posea el mayor Router ID será designado BDR.
5. Si uno o más routers de la lista de elegibles a DR se declara candidato a DR, aquel que tenga la mayor prioridad será declarado DR. En caso de empate aquel router que posea el mayor Router ID será designado DR.
6. Si ningún router se declara candidato a DR aquel que fue elegido como BDR será el nuevo DR.
7. Si el router que está realizando el proceso originalmente de elección es electo como el nuevo DR o BDR, o si deja de ser alguno de ellos, el proceso se repite desde el paso 2 hasta el 6.

2.8.4 Area Border Routers

Los ABRs (*Area Border Router*) son aquellos routers situados entre el backbone y una o más áreas de bajo nivel. Cada uno de estos routers debe tener al menos una interfaz conectada con el backbone y con cada una de las áreas con que está conectado. Este tipo de router se encarga de sumarizar las rutas de las áreas a las que está conectada hacia el backbone.

2.8.5 Internal Routers

Los IRs (*Internal Router*) se encuentran en las áreas de bajo nivel y sus interfaces sólo están conectadas con routers dentro de la misma área.

2.8.6 Backbone Routers

Los BRs (Backbone Router) se encuentran localizados, como su nombre lo indica, en el área 0 ó backbone. Estos routers deben tener al menos una interfaz conectada con otro router en el backbone. Los ABRs también pueden ser considerados routers de backbone.

2.8.7 Autonomous System Boundary Routers

Estos ASBRs (*Autonomous System Boundary Router*) son routers que funcionan como puerta al tráfico externo, inyectando rutas al dominio OSPF que fueron aprendidos externamente mediante otros protocolos como BGP o EIGRP.

En la Figura 2.24 se puede identificar que R2 es un ASBR, por definición R2, R3, R4 y R5 son BRs, R4 y R5 son ABRs y R3 es un IR.

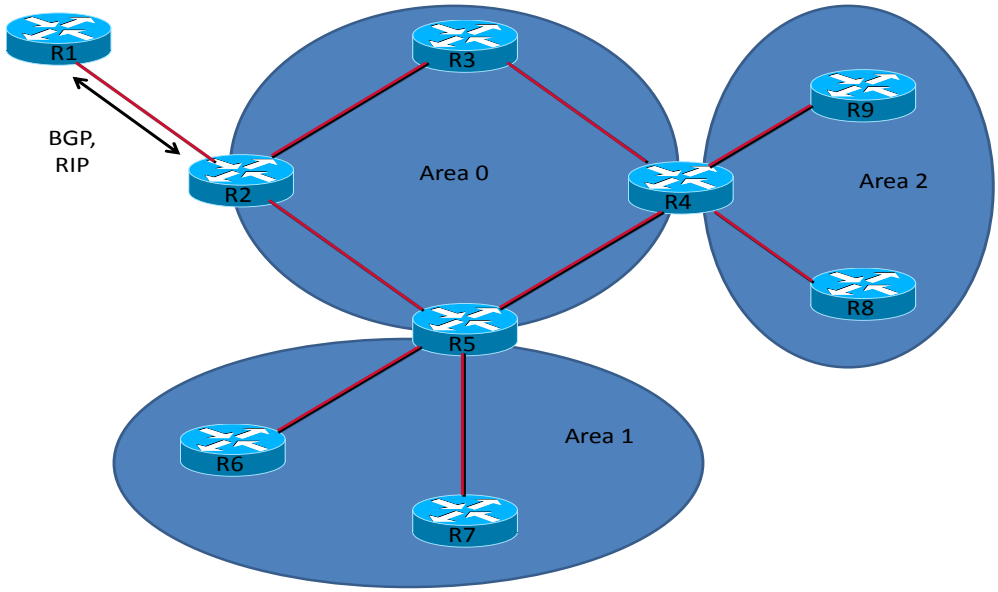

Figura 2.24: Tipos de Routers

2.9 Tipos de Redes

OSPF maneja 5 tipos de redes en su diseño:

- Punto-a-Punto: Se refiere a la conexión de dos routers directamente mediante una interfaz. Todos los routers OSPF conectados serán full adyacentes. Además se puede mencionar que en este tipo de redes no son utilizados los DR ni los BDR. Un ejemplo de estas redes serían aquellas que utilizan los protocolos PPP (*Point_to_Point Protocol*) [19] o HDLC (*High-Level Data Link Control*) [20].
- Broadcast: Se llama así a todas las redes conectadas mediante tecnologías como Ethernet o FDDI, por lo cual un nodo puede enviar un paquete y el mismo ser recibido por todos los otros nodos pertenecientes a la red. Las redes broadcast utilizan el concepto de DR y BDR.
- NBMA: Se refiere a aquellas redes que utilizan tecnologías como ATM (*Asynchronous Transfer Mode*) [21] o Frame Relay [3] donde dos o más routers pueden estar conectados sin capacidad de broadcast, por ende, cada paquete OSPF debe ser explícitamente enviado a cada router de la red, esto requiere una configuración extra para emular las redes broadcast. Aun con la característica de no poseer capacidad para manejar mensajes broadcast en este tipo de redes se utilizan los conceptos de DR y BDR.

- Punto-a-Multipunto: Son redes que no poseen capacidades de broadcast, sin embargo, su modo de operación es parecido a las redes punto-a-punto. Usualmente son redes orientadas a conexión como Frame Relay o ATM, pero a diferencia de estas es necesario que todos los routers de la red puedan comunicarse directamente.
- Virtual Links: Son usados un para conectar un área al backbone usando un área no backbone. Estos enlaces son configurados entre 2 routers de borde. También se usan estos enlaces si el backbone debe ser dividido a causa de una falla.

Se pueden introducir dos conceptos adicionales en cuanto a los tipos de redes soportados por OSPF:

- Red Stub: Es la red identificada con un prefijo IP que está conectada a un solo router dentro de un área.
- Red de Tránsito: Tiene dos o más routers adjuntos y llevan paquetes que sólo “pasan” por ella, es decir, ningún router de su red es el destino de dichos paquetes.

3 Evaluación de Proyectos Relacionados

Actualmente existen varias implementaciones de herramientas de enrutamiento que soportan los diversos protocolos que han sido desarrollados hasta el momento. Estas herramientas pretenden promover soluciones innovadoras de enrutamiento y seguridad. A continuación, se describen las más populares.

3.1 Zebra

Software con licencia GNU que maneja protocolos de enrutamiento basados en TCP/IP [3]. Soporta distintos protocolos como BGP-4, BGP-4+, RIPv1, RIPv2, RIPng [21] y OSPFv2.

Actualmente Zebra trabaja en plataformas como FreeBSD, NetBSD, OpenBSD y GNU/Linux sobre IPv4 e IPv6.

Una de las características principales de Zebra es que está hecho bajo un enfoque netamente modular donde cada protocolo tiene su propio demonio. Entre otras características que se pueden mencionar de Zebra se pueden mencionar:

- La tabla de enrutamiento se maneja a nivel de kernel.
- Maneja un demonio llamado Zebra que es el manejador de la tabla de enrutamiento del kernel y puede actuar como intérprete entre los diferentes demonios de protocolos implementados y dicha tabla de enrutamiento.
- Actúa como router dedicado.
- Posibilidad de cambiar dinámicamente la configuración de los demonios y poder observarlos mediante la interfaz de la terminal zebra.
- Cuenta con 2 modos de usuario para su administración. El modo normal donde el usuario puede ver el estado del sistema y el modo Enable donde se puede cambiar la configuración del mismo.

Actualmente Zebra no soporta protocolos multicast como BGMP, PIM-SM, PIM-DM los cuales se estarán incorporando en la versión Zebra 2.0[1]. Otras características que se podrán encontrar serían las de QoS, y filtros TCP/IP. En la Figura 3.1 se puede ver la arquitectura de Zebra.

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

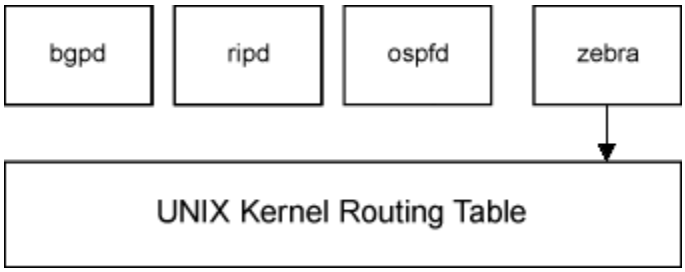

Figura 3.1: Arquitectura Zebra

Las plataformas soportadas por Zebra son:

- GNU/Linux 2.0.37
- GNU/Linux 2.2.x
- GNU/Linux 2.3.x
- FreeBSD 2.2.8
- FreeBSD 3.x
- FreeBSD 4.x
- NetBSD 1.4
- OpenBSD 2.5
- Solaris 2.6
- Solaris 7

Además Zebra soporta actualmente algunas pilas IPv6 como:

- Linux IPv6 stack for GNU/Linux 2.2.x y superiores.
- KAME IPv6 stack for BSD.
- INRIA IPv6 stack for BSD.

Entre los RFCs soportados por Zebra actualmente se pueden apreciar:

- 1058 (RIPv1).
- 2082 (RIP-2 MD5).
- 2453 (RIPv2).
- 2080 (RIPng).
- 2382 (OSPFv2).
- 2740 (OSPFv3).
- 1771, 2842, 2858 (BGP-4).

3.2 Quagga

Esta herramienta se puede definir como una suite de software de enrutamiento la cual provee implementaciones para OSPFv2, OSPFv3, RIPv1, RIPv2, RIPng, y BGP-4 para plataformas Unix[2], en particular, para FreeBSD, Linux, Solaris y NetBSD.

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

La arquitectura de Quagga consiste en una serie de demonios basados en la suite Zebra que actúa como una capa de abstracción para el kernel Unix que esté operando de manera subyacente y presenta el API "Zserv" a los clientes Quagga. Es Zserv quien implementa el protocolo de enrutamiento y comunica las actualizaciones al demonio Zebra.

Los Zserv implementados son los siguientes:

- OSPFD (Implementa OSPFv2).
- RIPD (Implementa RIPv1 y RIPv2).
- OSPF6D (Implementa OSPFv3).
- RIPNGD (Implementa RIPng).
- BGPD (Implementa BGPv4+).

Estos demonios son altamente configurables a través de una interfaz de comandos accesible vía red llamada "vty". También contiene una herramienta llamada "vtysh" que permite a los administradores tener monitoreados todos los aspectos de los demonios en un sólo lugar.

Entre otras características que se pueden mencionar de Quagga se pueden mencionar:

- La facilidad para agregarle nuevos demonios que implementen otros protocolos de enrutamiento.
- La interfaz interactiva que provee para el manejo de cada uno de los protocolos (Incluye vistas de tablas de enrutamiento).
- Quagga puede manejar cambios de direcciones de las distintas interfaces así como rutas estáticas.
- Soporta distintos modos de administración.
- El modo "normal" en el que sólo se puede observar el estatus del sistema.
- El modo "enable" en el cual es que se pueden hacer los cambios de configuración.

3.3 BIRD

BIRD[3] (*BIRD Internet Routing Daemon*) es un demonio de enrutamiento dinámico desarrollado por la Facultad de Matemática y Física de la Charles University en Praga. Está dirigido a sistemas Unix y distribuido bajo la licencia GNU. Soporta BGP, RIP, OSPFv2 y rutas estáticas.

BIRD se basa en una arquitectura modular que contiene los siguientes módulos:

- Módulos de Núcleo: Implementa las funciones del núcleo de BIRD como el manejo de las tablas de enrutamiento, mantenimiento del estatus de los protocolos, interacción con el usuario a través del CLI (*Command-Line Interface*).

[3] http://bird.network.cz

- Módulos de Librerías: Implementación de un amplio conjunto de librerías, funciones y utilidades.
- Módulos para Manejo de Recursos: Encargados de la administración de los recursos, colocación y liberación de ellos.
- Módulos de Configuración: Fragmentos de analizadores lexicográficos, reglas gramaticales. Para cada grupo de módulos existe un módulo de configuración.
- Filtros: Implementan el lenguaje de filtrado de los protocolos.
- Módulos de Protocolos: Implementan de manera individual los protocolos.
- Módulos Dependientes del Sistema: Implementan la interfaz entre BIRD y el Sistema Operativo.
- Cliente: Programa que provee un CLI amigable al usuario.

Entre las características más importantes de BIRD se pueden mencionar:

- Maneja un protocolo virtual para el intercambio de rutas entre diferentes tablas de enrutamiento en un único host.
- Establece un "BIRD client" como interfaz de configuración online para revisar el estatus del demonio.
- Fácil configuración.
- Herramientas robustas para el filtrado de las tablas de enrutamiento.
- Respuesta de eventos en tiempo real.
- La configuración inicial según el sistema operativo depende de una serie de scripts para la autoconfiguración.
- Está desarrollado con el lenguaje C.

3.4 OpenOSPFD

OpenOSPFD[4] es una implementación del protocolo OSPF distribuida de manera gratuita, la cual permite que computadores ordinarios sean usadas como routers. El diseño de OpenOSPFD está basado en otra implementación de los mismos diseñadores: llamada OpenBGPD. El demonio padre de OpenOSFPD está dividido en tres procesos:

- El motor OSPF maneja todo lo concerniente a los paquetes entrantes y al estado de la máquina con todos sus eventos y tiempos de espera, elección del DR y BDR. Además es responsable de la inundación confiable de los LS Updates así como su retransmisión y sus LS ACK correspondientes.
- Route Decision Engine: El RDE almacena la base de datos topológica, calcula el árbol SPT e informa al demonio padre de los cambios en la tabla de enrutamiento. También sincroniza múltiples áreas si el router está actuando como ABR.
- OSPFCTL: Es la herramienta para controlar y monitorear OpenOSPFD. Usa un socket local Unix para comunicarse con el demonio *ospfd.*

[4] http://www.openospfd.org

OpenOSPFD no implementa ninguna interfaz para su uso ya que se basa en el uso de las diferentes shells que contiene los sistemas operativos basados en Unix para poder controlar la aplicación.

3.5 XORP

XORP[5] está dividido en dos subsistemas. El nivel más alto (espacio de usuario) consiste en los protocolos de enrutamiento en sí y su manejo. El nivel más bajo (*kernel*) provee las rutas de transmisión y diferentes APIs para el acceso al espacio de usuario.

El nivel de usuario está basado en una arquitectura multi procesos con un proceso por protocolo de enrutamiento y un mecanismo inter-procesos llamado XRLs (*XORP Resource Locators*). La comunicación mediante los XRLs no está limitada a un solo host sino que puede implementarse de manera distribuida.

El nivel bajo usa un kernel de transmisión Unix, un kernel de transmisión Windows (Windows 2003 Server).

Entre otras características de XORP se pueden nombrar:

- La modularidad y la poca dependencia entre los 2 niveles provee muchas posibilidades en cuanto al motor de transmisión.
- Usa procesos separados para IPv4 e IPv6.
- Aunque implementa una arquitectura modular, creando un proceso para cada protocolo, es posible compilar varios de esos procesos para correr como uno solo.
- XORP provee una arquitectura flexible para los desarrolladores.

Una parte importante de XORP es el FEA (*Forwarding Engine Abstraction*) que provee una interfaz para el manejo de las funcionalidades básicas de los protocolos de enrutamiento.

También se tiene que mencionar otra parte esencial de XORP llamada RIB (*Routing Information Base*) que mantiene una copia en espacio de usuario de las tablas de enrutamiento. Este módulo se comunica con los protocolos para instanciar las rutas y también con el FEA para instalar las entradas apropiadas de transmisión. El RIB también mantiene información para rutas que utilizan multicast (*Multicas-Capable Routes, MRIB*) para ser usada como información por el RPF (*Reverse-Path Forwarding*).

En la Figura 3.2 se puede ver la arquitectura de XORP.

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

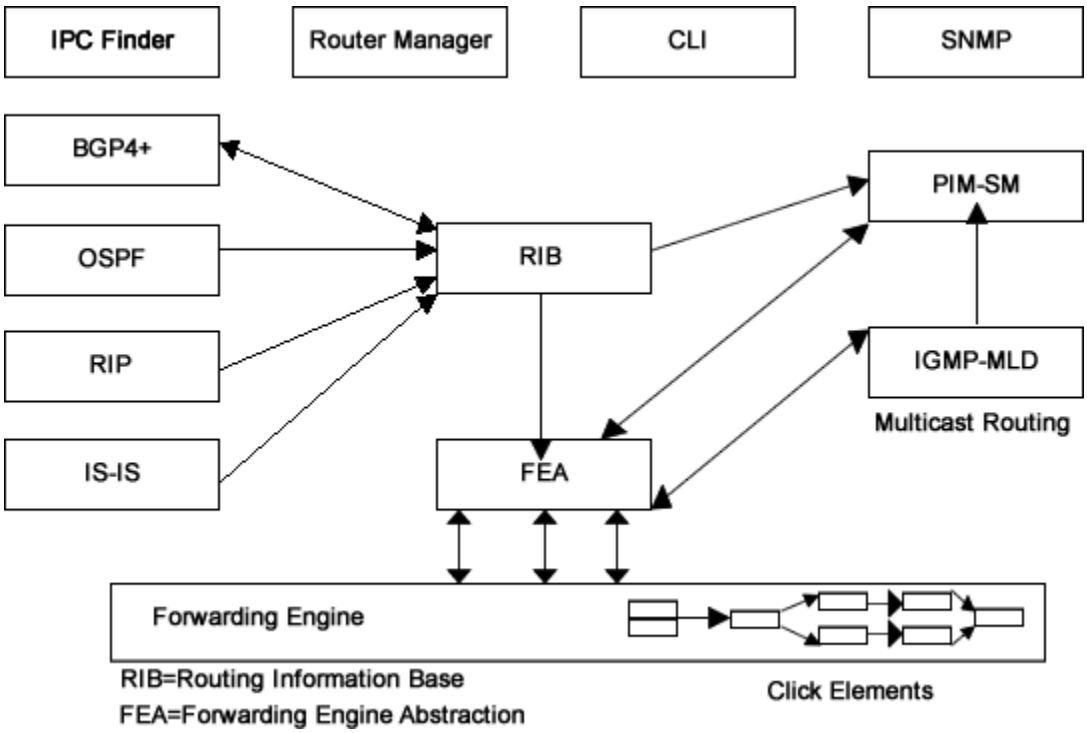

Figura 3.2: Arquitectura XORP

XORP provee procesos para los siguientes protocolos:

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

XORP es bastante portátil corriendo en FreeBSD, Linux, OpenBSD, DragonFlyBSD, NetBSD, MacOS X y hay incluso una adaptación para Windows Server 2003, que sólo es compatible con IPv4.

3.6 Routing And Remote Access en Windows Server 2003

Windows Server 2003 provee la capacidad de usar como routers computadores ordinarios a través del RRAS[6] (*Rounting and Remote Access*). Además de las capacidades de enrutamiento, permite configurar VPNs, servicios básicos de firewall y logs de procesos. Esta implementación de OSPF para Windows ofrece múltiples funcionalidades propias del protocolo. En la ventana de configuración principal se encuentran parámetros como: *area ID, router priority, cost* y *password*, como también elegir el tipo de red a la que se está conectado (broadcast, NBMA o point-to-point).

En la opción "advanced" de la ventana de configuración principal, se encuentran otros parámetros de OSPF como el hello interval, MTU, transit delay, dead interval, retransmit interval, poll interval. Dentro de las propiedades de esta implementación se encuentra lo correspondiente a las áreas ya que permite definirlas según las especificaciones del protocolo. Dentro de las posibilidades de configuración, el RRAS permite establecer el Router ID, así como habilitar al host como ASBR.

Es importante destacar que RRAS no implementa interfaces virtuales o "loopback" sólo trabaja con interfaces físicas del host y aunque en la configuración se puede apreciar la opción de "Virtual Interfaces" esto se refiere a una conexión lógica de punto a punto entre un ABR de un área y un ABR que está conectado físicamente al área de backbone.

3.7 Easy-EIGRP

Trabajo Especial de Grado realizado para la Universidad Central de Venezuela en Enero del 2010 el cual consiste en el desarrollo de una aplicación para la enseñanza del protocolo EIGRP (*Enhanced Interior Gateway Routing Protocol*) patentado por Cisco Systems. La aplicación provee un conjunto de herramientas que permiten depurar paso a paso todos los procesos llevados a cabo durante el funcionamiento de EIGRP en distintos escenarios [23] [24].

El trabajo se inicia con un estudio basado en ingeniería inversa, el cual consiste en el desarrollo de situaciones puntuales para el estudio del comportamiento del protocolo utilizando routers Cisco.

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

[6] http://technet.microsoft.com/en-us/network/bb545655.aspx

4 Marco Metodológico

Para la implementación de la aplicación se decidió utilizar una metodología Ad Hoc de tipo incremental, basada en iteraciones, que fueron divididas en cuatro etapas: análisis y planificación, diseño, codificación y pruebas. Se planificaron iteraciones para cada uno de los objetivos establecidos en un análisis general de la solución, derivando en un proceso de múltiples iteraciones en las que se aplicaron las fases anteriormente mencionadas, como será descrito en el Capítulo 6.

4.1 Adaptación de la Metodología de Desarrollo

Con el fin de satisfacer los objetivos del TEG se establecieron puntos estratégicos del protocolo para definir el desarrollo de la aplicación. Como se puede ver en la Tabla 4.1, estos puntos fueron divididos para ir atacando el problema con mayor facilidad.

Sincronización de los Routers	• Envío y recepción de paquetes. • Estados de los vecinos. • Estados de la interfaz de red. • Fases del protocolo. • Bases de datos topológica.
Interfaz de Usuario	• Configuración de las interfaces de red. • Graficación de la topología. • Sniffer OSPF. • Tablas de enrutamiento y vecinos.

Tabla 4.1: Puntos de Enfoque en el Desarrollo

A medida que avanzó la implementación del protocolo aparecieron necesidades no contempladas al inicio del desarrollo, por lo que se incluyeron nuevos focos en el proceso para garantizar la eficacia del mismo.

A cada uno de los puntos de enfoque se le aplicó una metodología basada en cuatro etapas: análisis, diseño, codificación y pruebas; así como también un desarrollo secuencial de cada uno de los puntos.

4.2 Análisis

En la fase de análisis se definen los requerimientos del punto a desarrollar, y a su vez la lista de sub requerimientos, obteniendo así una división del problema y una visión específica de lo que se debe realizar para cumplir los requisitos.

Se organizan estos requerimientos en un esquema jerárquico colocando los objetivos generales por encima de aquellos objetivos específicos identificados, para mantener un orden en el desarrollo de la aplicación.

4.3 Diseño

Esta etapa propone la estructura lógica de las soluciones a los requerimientos: definición de clases, argumentos y planes que integraran las partes del desarrollo que busca satisfacer un requerimiento puntual. Se realizaron diagramas de clases para ordenar las estructuras a utilizar, realizando un énfasis entre la relación en las mismas para simplificar la implementación del código para la aplicación Easy-OSPF.

4.4 Codificación

En esta fase se procede a la implementación de código necesario utilizando todas aquellas metodologías posibles, como el uso de patrones de diseño, así como la documentación del código.

4.5 Pruebas

Esta fase se basa en la búsqueda de aquellos casos en donde los requerimientos puedan no ser cumplidos debido a problemas en la codificación o el surgimiento de situaciones no contempladas. En el caso de encontrar fallos, los mismos son registrados y pasan a formar parte de nuevos requerimientos, a los que posteriormente se les aplicará de nuevo la metodología anteriormente descrita.

Estas pruebas, se producen a partir de un conjunto de datos de entrada y una topología de red en particular para validar que los datos de salida sean correctos. Los datos de entrada son todos aquellos relacionados a la interfaz o interfaces que maneja la aplicación en un momento dado.

Las pruebas siempre se realizan combinando la interfaz grafica y el código en la implementación del protocolo, para que los resultados de dichas pruebas sean los más generales posibles dentro del ámbito específico del requerimiento que se está tratando de cubrir.

Debido al desarrollo secuencial de los requerimientos la mayoría de las veces los resultados de una prueba son el punto de partida de la fase de análisis que se requerirá para la obtención de otro objetivo.

4.6 Tecnologías Utilizadas

A continuación se da una breve descripción de cada una de las herramientas utilizadas para desarrollar Easy-OSPF:

- C#: Lenguaje de programación orientado a objetos, desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por ECMA e ISO [22]. Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma

.NET, similar al de Java aunque incluye mejoras derivadas de otros lenguajes.

- SharpPcap[1]: Framework de captura y envío de paquetes para .NET, basada en WinPcap. Su propósito es proveer un API para captura, inyección, análisis y construcción de paquetes utilizando cualquier lenguaje .NET.
- Devcon[2]: Herramienta basada en línea de comandos que provee las funcionalidades del Administrador de Dispositivos (*Device Manager*). Gracias a esta herramienta, es posible habilitar, deshabilitar, reiniciar, actualizar, remover y hacer consultas sobre un dispositivo específico o sobre un grupo de ellos.
- Piccolo2D[3]: Librería que brinda un conjunto de herramientas para el desarrollo de programas gráficos en 2D usando ZUIs (*Zoomable User Interface*) la cual es un nuevo paradigma de interfaces que representan un canvas de gran tamaño en un display tradicional.
- PcapDotNet[4]: Es un *wrapper* de WinPcap para .NET escrito en C++/CLI y C#. Provee casi todas las características de WinPcap e incluye un framework de interpretación de paquetes.
- Visual Studio 2008: Entorno de desarrollo integrado para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.
- Log4Net[5]: Librería que proporciona una serie de herramientas para el desarrollo de archivos de logs. Permite una variedad amplia de tipos de archivos de destino.

4.7 Prototipo General de la Interfaz

Para darle al usuario mayor poder de navegación entre los distintos módulos de la aplicación se usó un modelo de pestañas para posicionarlos. La ventana principal se dividió de tal forma que mostrara la mayor cantidad de campos configurables posibles. Las dimensiones, en pixeles, usadas para el diseño de la interfaz principal fueron 950 de ancho por 680 (Figura 4.1).

[1] http://www.tamirgal.com/blog/page/SharpPcap.aspx
[2] http://support.microsoft.com/kb/311272
[3] http://www.piccolo2d.org
[4] http://pcapdotnet.codeplex.com
[5] http://logging.apache.org/log4net

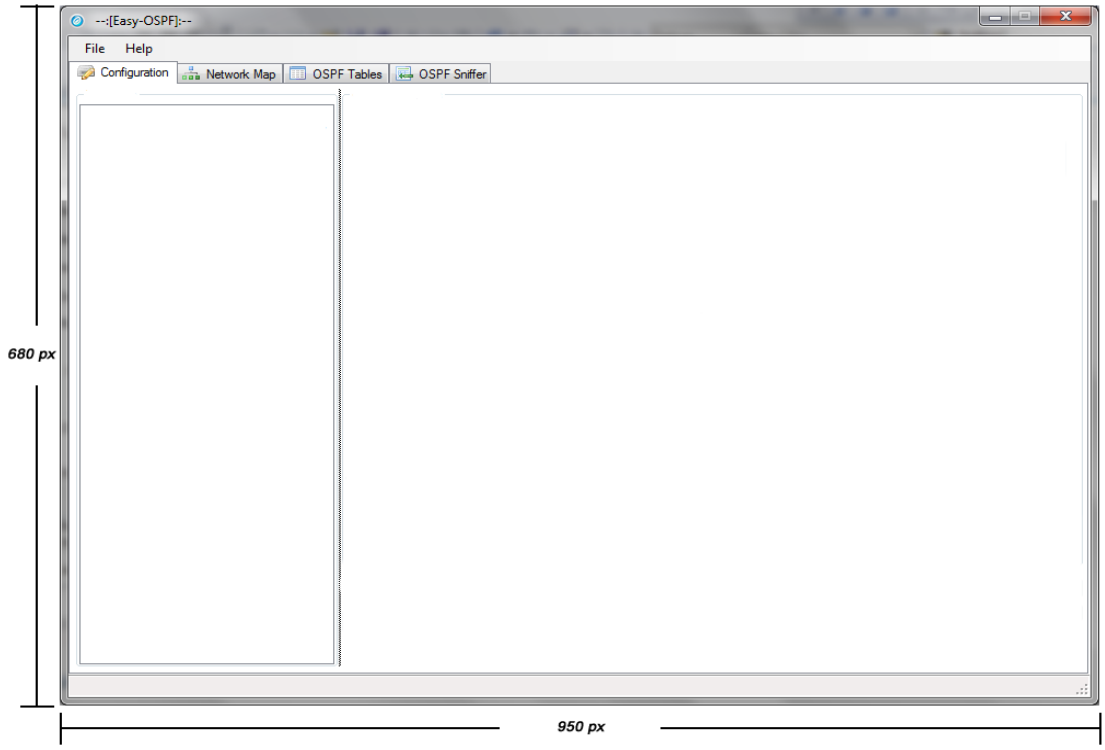

Figura 4.1: Modelo General de Interfaz

Para las divisiones internas, en los distintos módulos a implementar se usaron *Group Box*, componente de C# que permite definir un marco a un grupo de controles establecidos.

5 Marco Aplicativo

La metodología que se definió en el Capitulo 4 se aplica de manera estricta sobre una serie de objetivos que surgieron del análisis general de la situación, en donde se procederió a determinar las necesidades y la forma de solucionarlas.

5.1 Implementación del Protocolo

La implementación del protocolo OSPF en Easy-OSPF se basó en RFC 2328 tomando en cuenta los detalles en el diseño de los métodos descritos en el.

La base de Easy-OSPF se encuentra en la clase *ThreadListenPacket* la cual es la encargada de recibir todos los paquetes del protocolo y enviar la mayoría de las respuestas adecuadas a dichos paquetes. Además controla y ejecuta todas las acciones que toma OSPF con respecto a la llegada de cada uno de los tipos de paquetes. En esta clase se desarrollan los cambios entre los diferentes estados que puede tomar una interfaz OSPF.

La clase *fPrincipal* es donde se encuentra todo el manejo de la configuración de interfaces de red disponibles para la configuración del usuario por parte del usuario. También se manejan los diversos estados de operación en los cuales una interfaz puede encontrarse (enable,passive o disable). Además se implementan en dicha clase todos los métodos de validación para los parámetros de configurables que aparecen en Easy-OSPF. Igualmente se debe mencionar que en la clases *fPrincipal* se verifican todos aquellos cambios externos relativos a las interfaces de red, es decir, cambios de IP, inhabilitación de interfaces etc, a nivel de Windows. Una de los aspectos más importantes de esta clase es que maneja todas las interfaces de Easy-OSPF a nivel gráfico, iniciando los procesos de graficación para el módulo *Netowrk Map* e insertando los paquetes en la interfaz de los módulos OSPF Tables y OSPF Sniffer.

Otra de las clases importantes en la implementación de Easy-OSPF es la clase *InterfaceSettings* que es aquella que toma toda la información de las interfaces configuradas y se las proporcionada a las demás clases involucradas en el funcionamiento de OSPF.

En cuanto al envío de paquetes estos se manejan de dos maneras:

- Para los paquetes Hello y DBD Description de sincronización inicial se implementaron 2 hilos llamados *HelloThread* y *DBDSyncThread* los cuales se encargan de enviarlos cada *Hello Interval* en el caso de los paquetes Hello y cada vez que empiecen una sincronización de bases de datos topológicas en el caso de los paquetes DBD.

- En cuanto a los diferentes LSAs se implementaron clases para cada tipo de ellos en las cuales se realizaban las tareas de fraccionamiento y clasificación de la información contenida en los paquetes entrantes y de armar un paquete conteniendo la respuesta a dicho paquete.

En la clase *TopologicDB* se encuentran los métodos concernientes al manejo de la base de datos topológica, es decir, aquellos métodos que verifican cuando un LSA puede agregarse o borrarse de la base de datos. Además maneja el cálculo de la tabla de enrutamiento implementando Dijkstra. También cabe destacar que en esta clase se manejan todos los timers que especifica OSPF para mantener actualizada la base de datos topológica.

El módulo Network Map utiliza principalmente la clase *GraphEditor* para manejar la graficación de la topología tratando a los routers como nodos de un grafo definido por los routers LSA y networks LSA que constituyen la base de datos topológica. En el RFC que describe OSPF se establece que toda aquella información correspondiente a un router que ya no está activo se mantiene en la base de datos por espacio de una hora lo cual presentó la mayor traba a la hora de implementar la graficación, por lo cual se implementaron una gran cantidad de validaciones para definir los routers a graficar correctamente.

Los eventos concernientes a los routers directamente adyacentes a Easy-OSPF o routers vecinos, la clase *Neighbor* es aquella encargada de manejarlos enviando los LSAs, y tomando las acciones necesarias, correspondiente ante la ocurrencia de cualquier evento asociado.

También se manejan algunas estructuras de datos (listas) con las cuales se manejan recursos críticos. Entre estas listas se deben mencionar *ListofNeighbors* que es una lista de instancias de la clase *Neighbor* e *InterfaceList* que una lista de instancias de la clase Interface.

5.2 Análisis General

Para el comienzo del desarrollo de la aplicación se llevó a cabo un análisis previo para establecer puntos importantes que serán los primeros requerimientos a satisfacer. Los puntos establecidos en esta fase de análisis fueron son los siguientes:

- Diseño de una interfaz gráfica.
- Implementación del envío y recepción de paquetes.
- Definición de los campos de configuración por parte del usuario.
- Establecimiento de los fundamentos de la graficación para la base de datos topológica.
- Definición de la estructura del Sniffer/Logger.
- Implementación del SPT (Shortest Path Tree).
- Definición de las estructuras a utilizar para la base de datos topológica, tabla de vecinos y tabla de enrutamiento.

5.3 Desarrollo de la Aplicación

Easy-OSPF consiste de 4 módulos:

1. Configuración de Interfaces.
2. Topología.
3. Sniffer OSPF.
4. Tablas de enrutamiento, base de datos topológica y tabla de vecinos.

La relación entre los módulos se puede observar en la Figura 5.1, donde se muestra el diagrama general de clases de la aplicación.

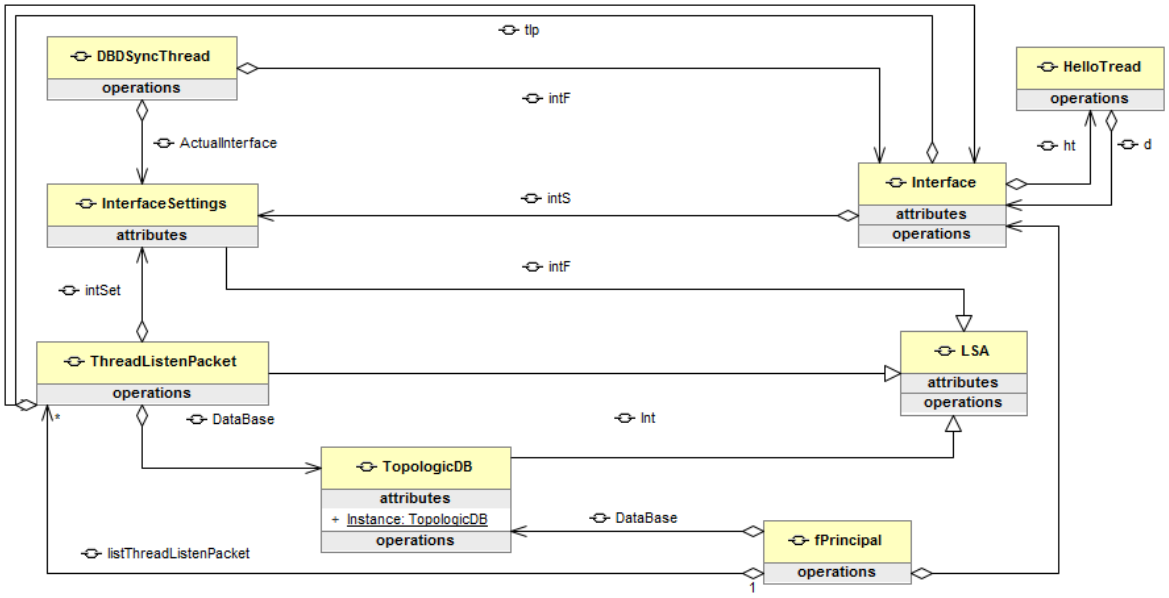

Figura 5.1: Diagrama de Clases General

A continuación se explicará el desarrollo de cada uno de los módulos de Easy-OSPF.

5.4 Módulo 1: Configuration

Módulo principal de la aplicación donde se encuentran los parámetros configurables del protocolo y de las interfaces de red disponibles.

5.4.1 Fase de Análisis

Para el desarrollo de este módulo se tomó como principal objetivo el mostrarle al usuario todas aquellas interfaces disponibles para ser configuradas bajo OSPF. Además se le debe de proveer todos aquellos parámetros configurables por cada una de las interfaces.

5.4.2 Fase de Diseño

Ya que el módulo *Configuration* es la interfaz gráfica con que el usuario cuenta para configurar el protocolo OSPF, las clases que manejen este módulo deben ser las encargadas de capturar dicha información y transferirla a las clases definidas para ordenarla y posteriormente hacer uso de ella.

Como se puede apreciar en la Figura 5.2, la clase que implementa la interfaz principal, *fPrincipal*, se relaciona especialmente con la clase *InterfaceList,* la cual es la encargada de algunas tareas como: registrar la información que viene desde la interfaz de usuario en estructuras creadas para ello, manejar algunos hilos de ejecución importantes en el proceso de sincronización como el *HelloThread* (encargado de enviar los mensajes Hello) y el *ThreadListenPacket* que es el corazón de la sincronización en la aplicación al ser el encargado de procesar los paquetes que lleguen a esa interfaz.

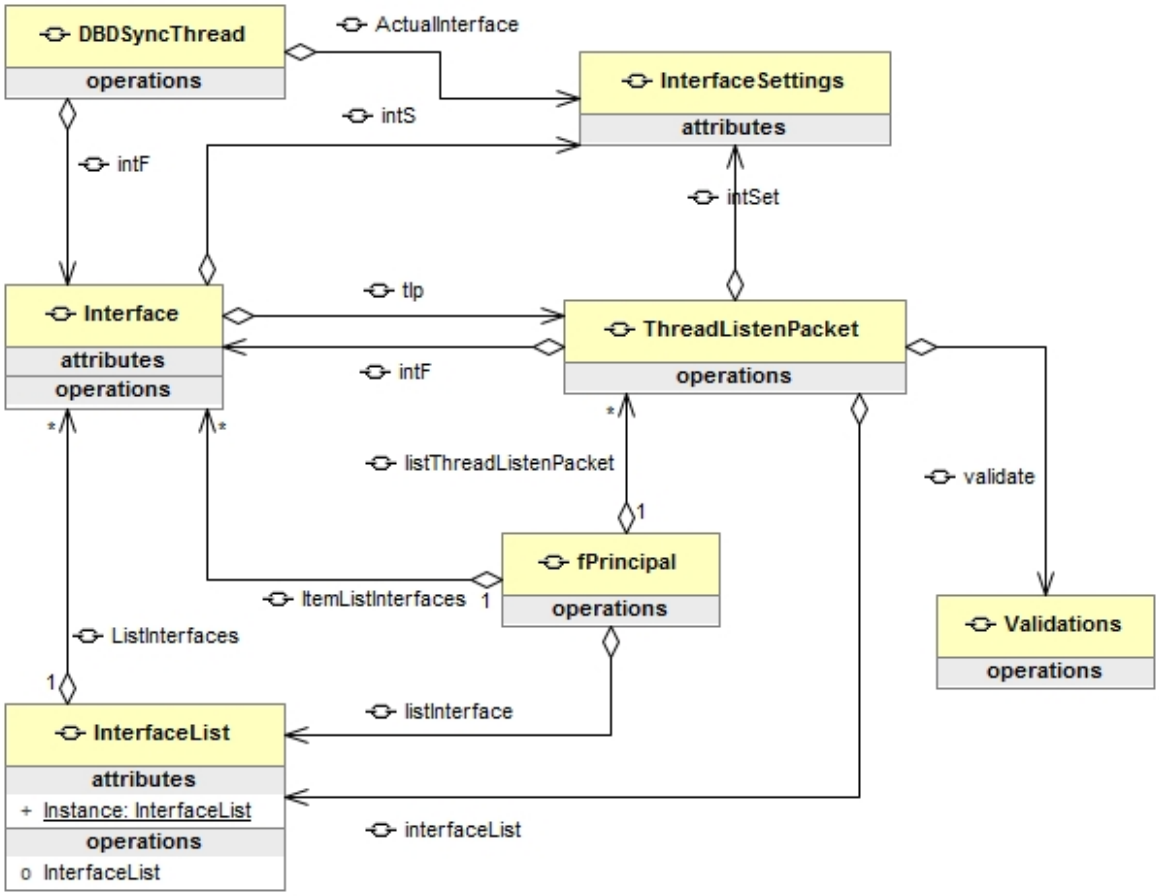

Figura 5.2: Diagrama de Clases del Módulo Configuration

5.4.3 Fase de Codificación

Con el módulo OSPF Settings el usuario cuenta con una interfaz gráfica dedicada a la configuración del protocolo OSPF en las interfaces disponibles. A nivel de codificación se suministraron los campos necesarios para dicha configuración. Además, se desarrollaron métodos para validar los campos que así lo requieran (Figura 5.1). Los parámetros de configuración por interfaz se dividieron en 4 grupos, que se describen a continuación:

1. Interface Settings: Parámetros relativos a la interfaz de red más no al protocolo OSPF.
   - IP Address: Dirección IP de la interfaz de red, en este caso debe validarse que la dirección IP no se encuentre duplicada en la red y que tenga un formato válido para IPv4.
   - Network Mask: Máscara de subred de la interfaz de red. Las validaciones necesarias incluyen formato adecuado y coherencia con respecto a la dirección IP.
   - Metric: Indica el gasto requerido para enviar paquetes a través de la interfaz seleccionada, se debe validar la coherencia en el valor.
   - Hello Interval: Intervalo en segundos que debe haber entre dos paquetes Hello.
   - Dead Interval: Tiempo en segundos que se esperara antes de declarar a un vecino como caído.

2. Interface Mode: Modo en el cual va a trabajar la interfaz.
   - Enable: Interfaz configurada bajo OSPF, en este estado es capaz de establecer adyacencias con otras interfaces OSPF que pertenezcan a routers del sistema autónomo.
   - Passsive: Interfaz en modo “promiscuo”, es decir, solo es capaz de escuchar los paquetes que viajan a través del medio.
   - Disable: Interfaz deshabilitada para cualquier intercambio de información.

3. OSPF Interface Settings: Parámetros relacionados a una interfaz OSPF activa.
   - Router Priority: Indica la prioridad que va a tener un router a la hora de elegir el Designated Router y el Backup Designated Router.

4. OSPF Interface Settings: Parámetros relativos al protocolo OSPF.
   - Router ID: Identificador unívoco del router en el sistema, por conveniencia se tomó este identificador con el mismo formato de una dirección IPv4, por ende se debe validar el formato de la misma.
   - Area ID: Identificador del área OSPF donde se está configurando la interfaz. Debido a los alcances de este TEG, este identificador no será modificado y siempre tendrá el valor “0.0.0.0”, es decir backbone.

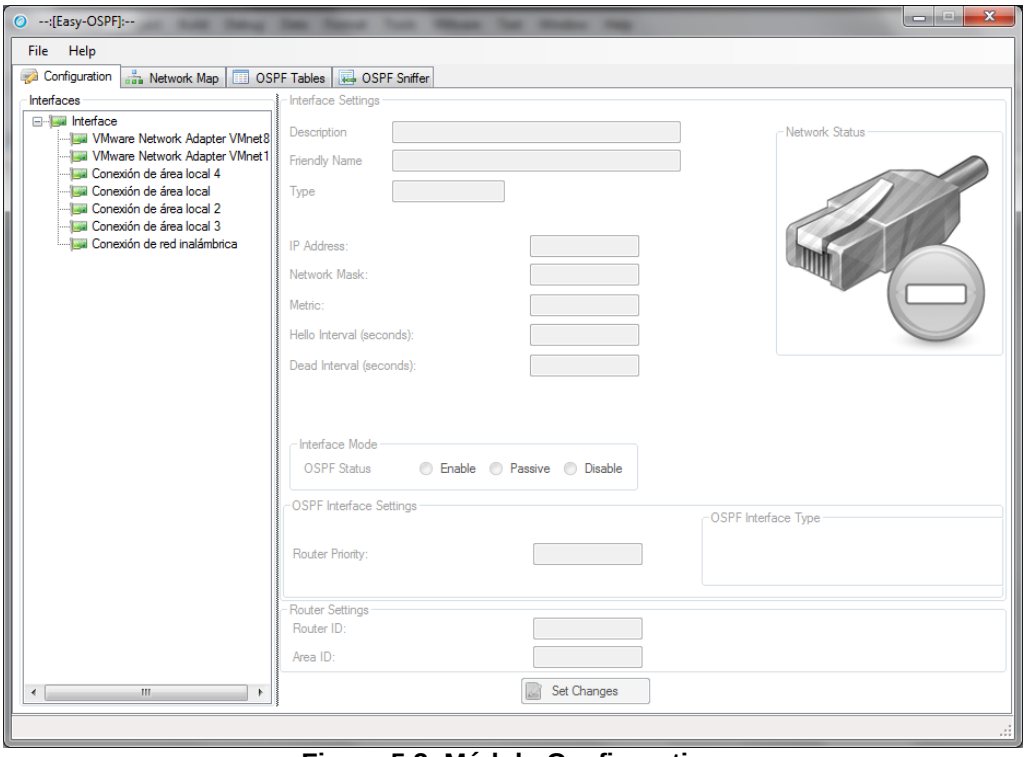

Figura 5.3: Módulo Configuration

5.4.4 Fase de Pruebas

Las pruebas para este módulo se centraron en 2 puntos: la barra de menús y los campos para la configuración del protocolo OSPF.

Para la barra de menús, se verificó que las acciones que se describen en cada uno de los menús sea aquella para la cual fue establecida. Como se puede ver en Figura 5.3, la interfaz cuenta con los siguientes menús:

1. File:
   - *Save Config*: Guarda la configuración actual en un archivo XML.
   - *Load Config*: Abre un archivo de configuración en formato XML anteriormente guardado.
   - *Exit*: Sale de la aplicación.
2. Help: Proporciona la ayuda necesaria al usuario para el manejo de la aplicación.

Para los campos de configuración del protocolo simplemente se le colocaron valores extremos para conocer la respuesta de la aplicación a ellos.

5.5 Módulo 2: Network Map

Este módulo proporciona de forma gráfica la topología completa del sistema autónomo donde se encuentra la aplicación en tiempo real gracias a la sincronización de las bases de datos con los diferentes routers OSPF.

5.5.1 Fase de Análisis

El desarrollo de este módulo se basó en disponer de una manera eficiente la topología de red ya que esta puede llegar a tener amplias dimensiones. Además cuenta con una serie de herramientas para la manipulación de la misma como la capacidad de desplazar los routers mediantes *Drag and Drop*, exportar la topología a formato PNG, hacer Zoom In y Zoom Out a la topología, entre otros.

5.5.2 Fase de Diseño

La clase *GraphEditor,* parte de la librería *Piccollo2D*, es la responsable de la graficación, puesto que a través de ella se puede utilizar el conjunto de herramientas dispuestas para el proceso de dibujo de la topología. Esta clase se relaciona directamente tanto con la *fPrincipal*, puesto que de ahí recoge toda la información de las interfaces, como la clase *TopologicDB,* la cual le provee parte de la información necesaria para graficar la topología (Figura 5.4).

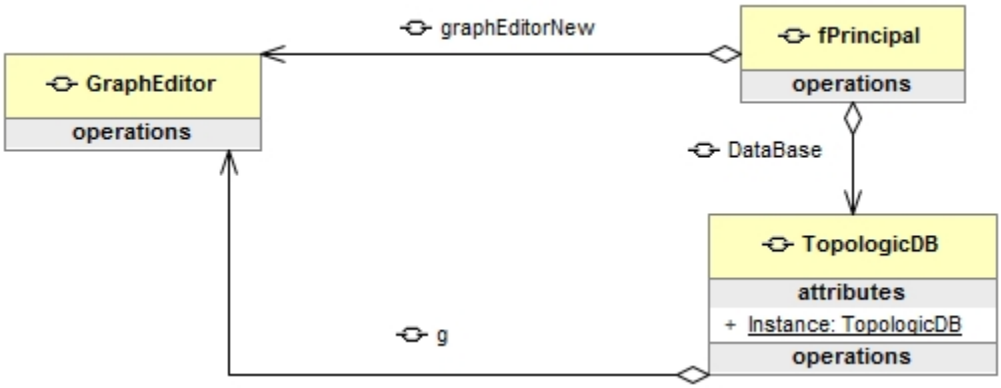

Figura 5.4: Diagrama de Clases del Módulo Network Map

5.5.3 Fase de Codificación

En este módulo, se utilizó esencialmente la librería Piccolo2D que brinda un conjunto de herramientas para el desarrollo de programas gráficos en 2D en ambientes .NET como C#. Se debe tener en cuenta que este módulo mantiene total dependencia de la base de datos topológica, así como de la tabla de vecinos de la aplicación ya que de ambas se desprende toda la información necesaria para realizar la graficación. En la Figura 5.5 se puede observar que la interfaz gráfica de este módulo consta de seis partes las cuales se describirán a continuación:

- OSPF Topology: Panel donde se muestra la topología de manera gráfica.
- Zoom: Controles para el Zoom In y Zoom Out de la imagen.
- Drag/Piking Mode: Controles para el Drag and Drop de la imagen.
- Save Topology: Guarda la topología en formato PNG.
- Help on Drag Mode: Ayuda.

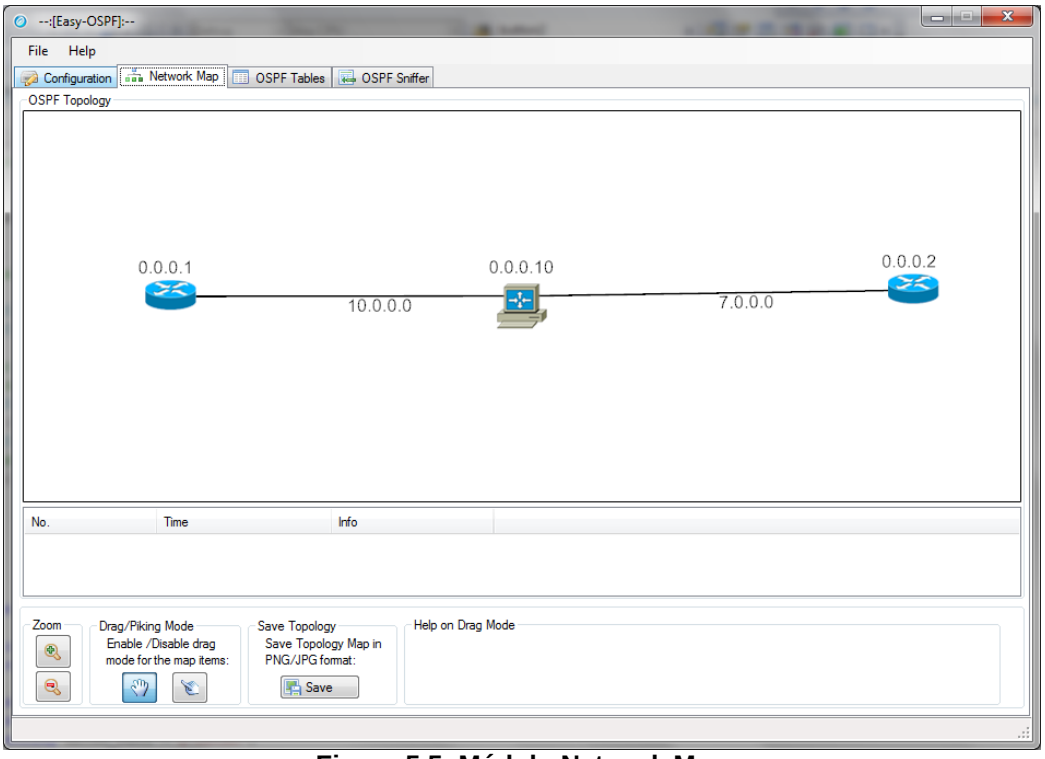

Figura 5.5: Módulo Network Map

5.5.4 Fase de Pruebas

Las pruebas realizadas son netamente orientadas a la formación de diferentes topologías, utilizando routers Cisco o el emulador de routers GNS3 (Graphical Network Simulator)[1] en conjunto con la aplicación para verificar que se graficaban correctamente. Estas pruebas serán explicadas con mayor profundidad en la Sección 5.8.

5.6 Módulo 3: OSPF Tables

En el módulo aquí descrito se procede a desplegar la información contenida en la tabla de vecinos, tabla de enrutamiento y en la base de datos topológica. Cada una de estas estructuras se puede ver individualmente mediante el selector ("Select a Table").

5.6.1 Fase de Análisis

El desarrollo de este módulo fue orientado a la exactitud que debe haber al mostrar la información de las tablas en la aplicación, ya que toda nueva información que se obtenga de la sincronización de bases de datos o del *flooding reliable* de OSPF debe ser reflejada inmediatamente.

[1] http://www.gns3.net

5.6.2 Fase de Diseño

Como se propuso anteriormente se debe mostrar de manera clara el contenido de los LSAs que forman la base datos topológica, así como toda aquella información de los vecinos cuando se hace referencia a la tabla de vecinos.

Se puede apreciar como la clase *TopologicDB* se relaciona con *fPrincipal* para mostrar todos los LSAs válidos que Easy-OSPF instaló en su base de datos topológica, los cuales se captan a través de la clase *ThreadListenPacket.*

En la tabla *Neighbor* nuevamente la clase *fPrincipal* muestra toda la información en un panel especificado para la misma. Esta información es capturada a través de la clase *ThreadListenPacket* y almacena en una estructura llamada *ListOfNeighbors* que contiene instancias de la clase *Neighbor* (Figura 5.6).

Por último, la tabla IP Routing Table se relaciona con la clase *TopologicDB*, ya que ella contiene los diferentes métodos para calcular el SPT utilizando Dijkstra tal como dicta el protocolo OSPF, y la clase *fPrincipal* ya que esta última es la encarga de mostrar la información en el panel de la aplicación.

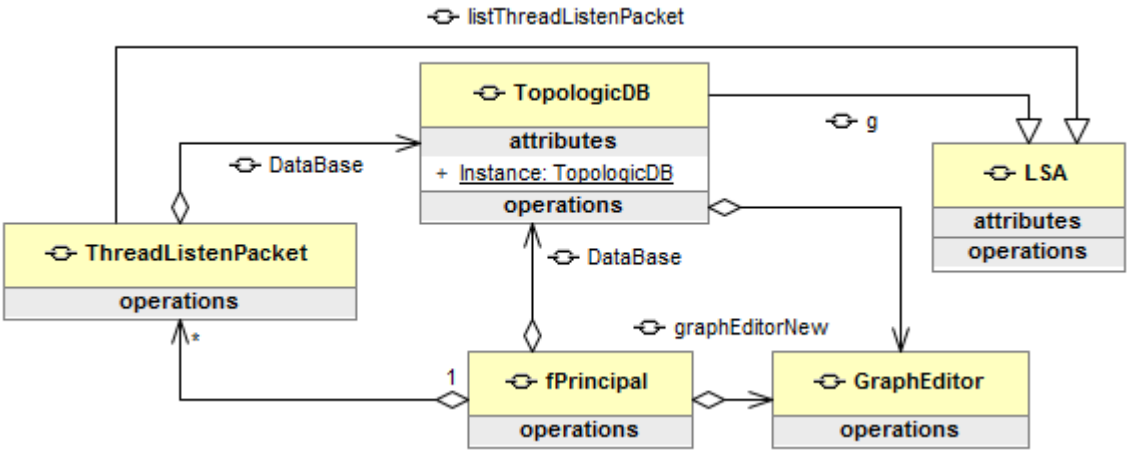

Figura 5.6: Diagrama de Clases del Módulo OSPF Tables

5.6.3 Fase de Codificación

La forma de desplegar las tablas se describe seguidamente:

- OSPF Database: Los LSAs obtenidos de los dos procesos anteriormente mencionados deben mostrarse entera y explícitamente al usuario, además de forma sencilla (Figura 5.7). Se sigue la estructura de los LSAs para desplegar la información en la interfaz.
- Neighbor: Se muestra la información siguiendo el mismo patrón mostrado por el CLI (*Command Line Interface*) de los routers Cisco, los cuales fueron utilizados en el proceso de desarrollo y pruebas (Figura 5.8).
- IP Routing Table: Igualmente se utilizó el patrón con el cual los routers Cisco, a través del CLI, muestran la información concerniente a sus tablas de enrutamiento (Figura 5.9).

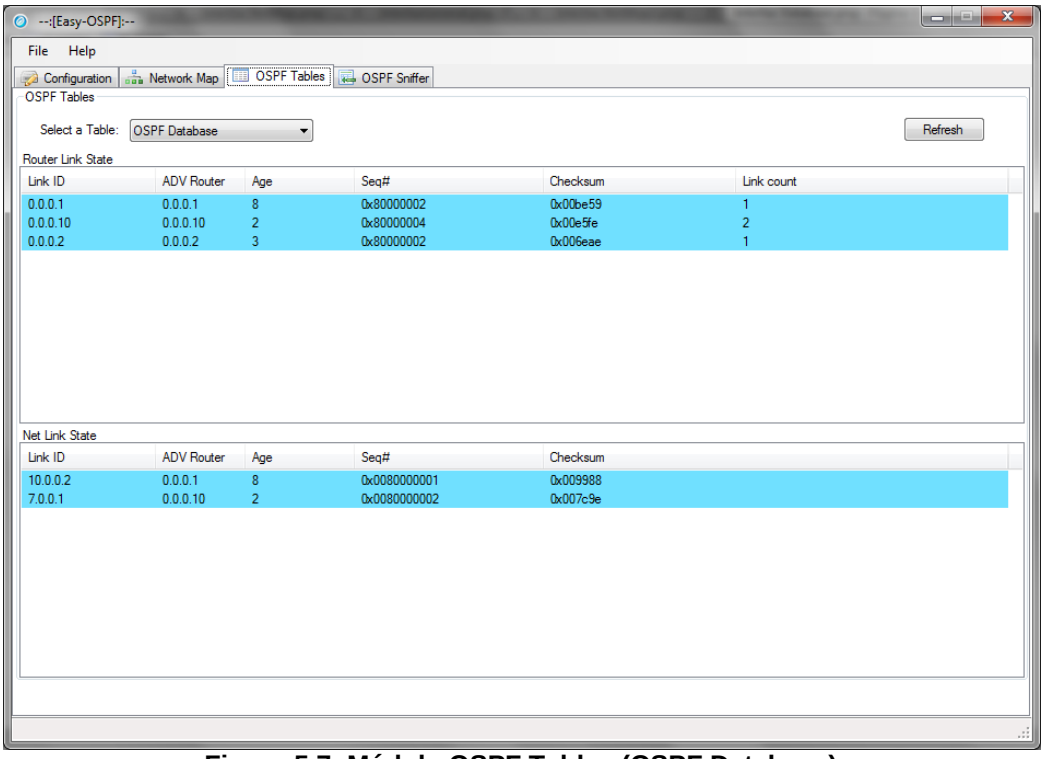

Figura 5.7: Módulo OSPF Tables (OSPF Database)

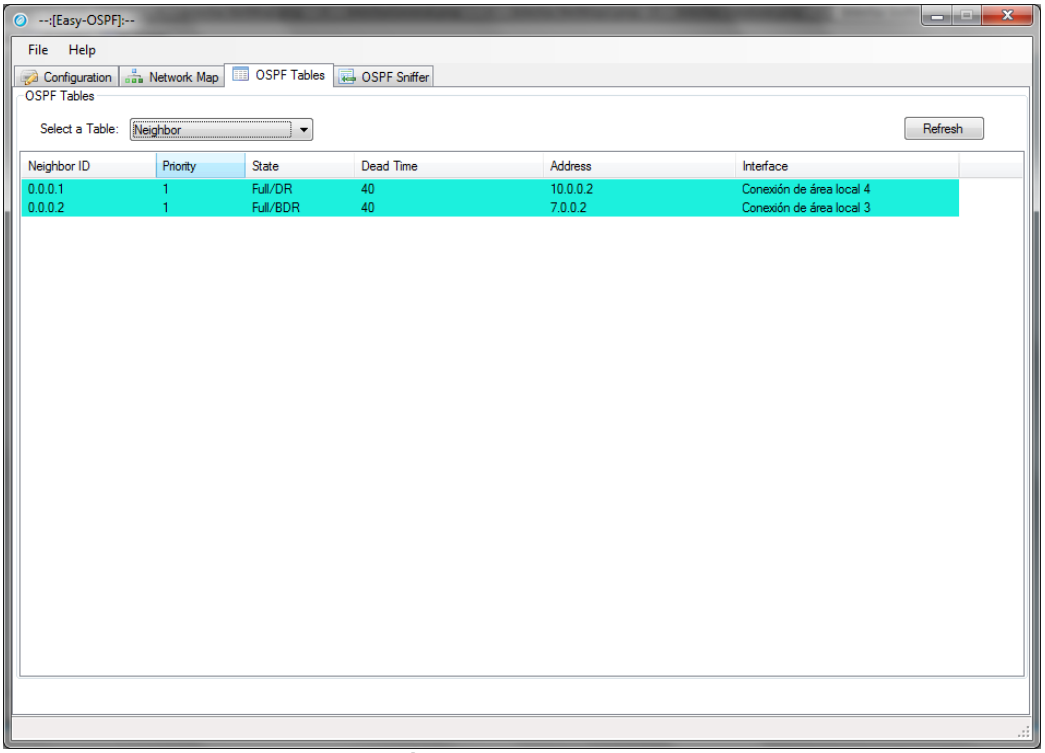

Figura 5.8: Módulo OSPF Tables (Neighbors)

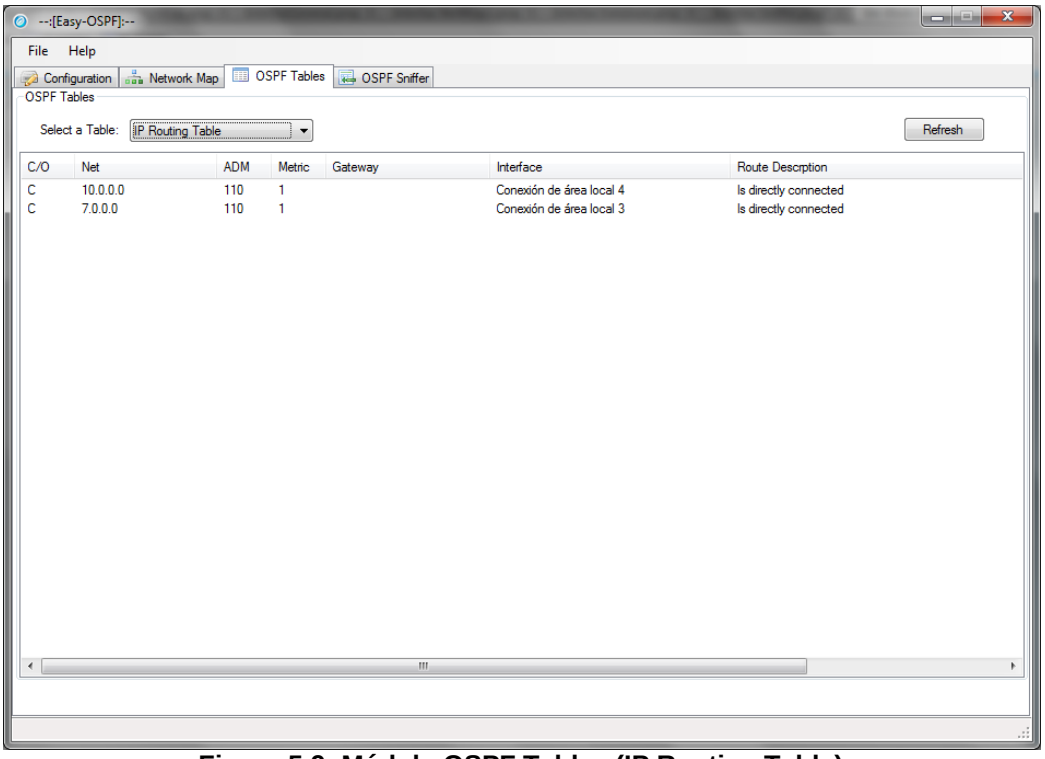

Figura 5.9: Módulo OSPF Tables (IP Routing Table)

5.6.4 Fase de Pruebas

Se deben especificar las pruebas para cada una de las tablas pertenecientes al módulo OSPF Tables, lo cual se muestra a continuación:

- OSPF Database: Se realizaron distintas pruebas de sincronización inicial de bases de datos topológicas, flooding, caída y levantamiento tanto de interfaces como de vecinos entre Easy-OSPF y routers GNS3, para así comparar sus bases de datos topológica ya que todos deben tener una copia exacta.
- Neighbor: Para esta tabla simplemente se diseñaron múltiples topologías para asegurar que registraba a todos los vecinos activos y sus correspondientes estados (DR, BDR, o Drother).
- IP Routing Table: Igualmente se diseñaron varias topologías en conjunto con routers GNS3, pero esta vez orientadas a mantener múltiples rutas para llegar a diversas interfaces, ya que así se podría corroborar el buen funcionamiento del SPT y a su vez la veracidad de la información mostrada en la tabla.

5.7 Módulo 4: OSPF Sniffer

En este módulo se contempló el desarrollo de un sniffer única y exclusivamente para paquetes OSPF para darle al usuario una vista general del proceso de intercambio de paquetes en la sincronización de bases de datos topológicas, descubrimiento de vecinos y otros procesos OSPF.

5.7.1 Fase de Análisis

En el desarrollo de este módulo se tomó en cuenta el nivel de detalle requerido para mostrarle la mayor cantidad de información al usuario de los paquetes OSPF que intervienen en el protocolo. Por ende antes que nada se decidió que cada paquete contara con un despliegue independiente de su información interna a manera de árbol, además de una vista hexadecimal del mismo.

5.7.2 Fase de Diseño

En la vista del módulo OSPF Sniffer se puede observar inicialmente el panel principal, donde son mostrados los paquetes OSPF que se están recibiendo /enviando. Para esto la clase *ThreadListenPacket*, encargada de recibir todos los paquetes a través de las interfaces y se relaciona con *fPrincipal* para mostrar la información a través de la interfaz dispuesta para ello. Ya que el módulo OSPF Sniffer también muestra los paquetes enviados por la aplicación se debe relacionar *fPrincipal* con la clase *Neighbor*, ya que esta produce paquetes al desaparecer un vecino, así como con la clase *HelloThread*, puesto que esta última clase es la encargada de mandar los paquetes Hello necesarios en el protocolo (Figura 5.10).

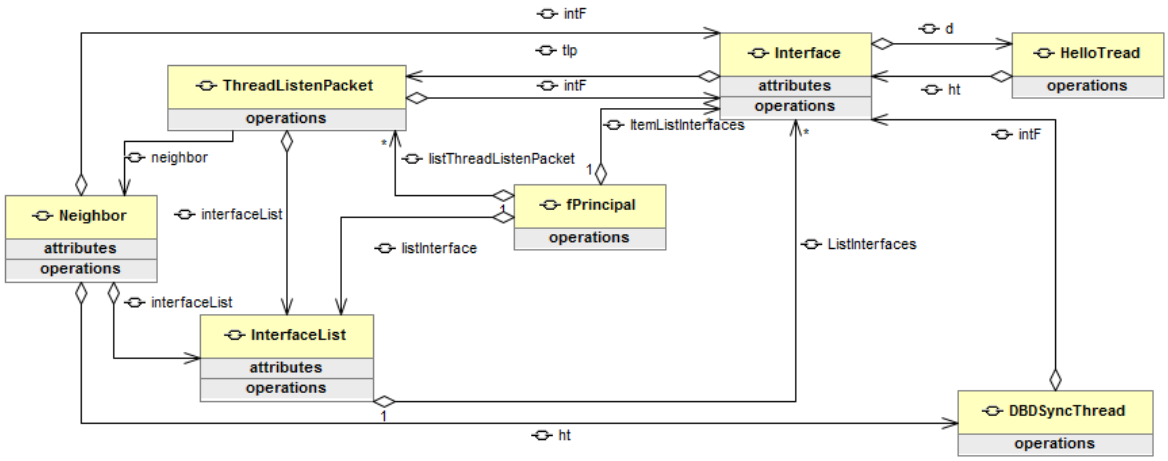

Figura 5.10: Diagrama de Clases del Módulo OSPF Sniffer

5.7.3 Fase de Codificación

El panel de este módulo cuenta con 6 columnas descritas a continuación (Figura 5.11):

- No: Número de paquete en orden ascendente o descendente según se prefiera.
- Time: Tiempo en que el paquete arribó desde el inicio de la aplicación.
- Source: Dirección IP fuente del paquete.
- Destination: Dirección IP destino del paquete.
- Packet Type: Se muestra el tipo de paquete OSPF que ha arribado.
- Processed Through: Interfaz por la que fue recibido el paquete.

Debajo del panel anteriormente descrito se encuentra otro panel dividido en dos. En el panel de la izquierda se muestra la información que contiene el paquete

OSPF seleccionado por el usuario, esta información será desplegada en forma de árbol. En el panel derecho se mostrara el contenido hexadecimal de dicho paquete.

En la parte inferior de la ventana están todos los controles para filtrar la información que se muestra en el sniffer:

- Debug, Info, Warn: Solo se muestran los paquetes del Debugger, de información o de warning según seleccione el usuario.
- OSPF Packets: Filtra según el tipo de paquete OSPF.
- Specific Filter: Muestra aquellos paquetes que cumplan una regla definida por el usuario.
- Interface Filter: Solo muestra los paquetes que son recibidos/enviados por la interfaz seleccionada.

La interfaz gráfica le da la libertad al usuario de combinar todos estos tipos de filtros para que sea mostrada una información determinada, es decir, aquellos paquetes que el usuario requiera en un momento específico. También se deben mencionar dos herramientas dispuestas en la interfaz las cuales son:

- Auto Scroll: Si está habilitada esta opción, el scroll del panel que contiene los paquetes baja automáticamente.
- Delete: Borra el contenido del panel que muestra los paquetes OSPF.

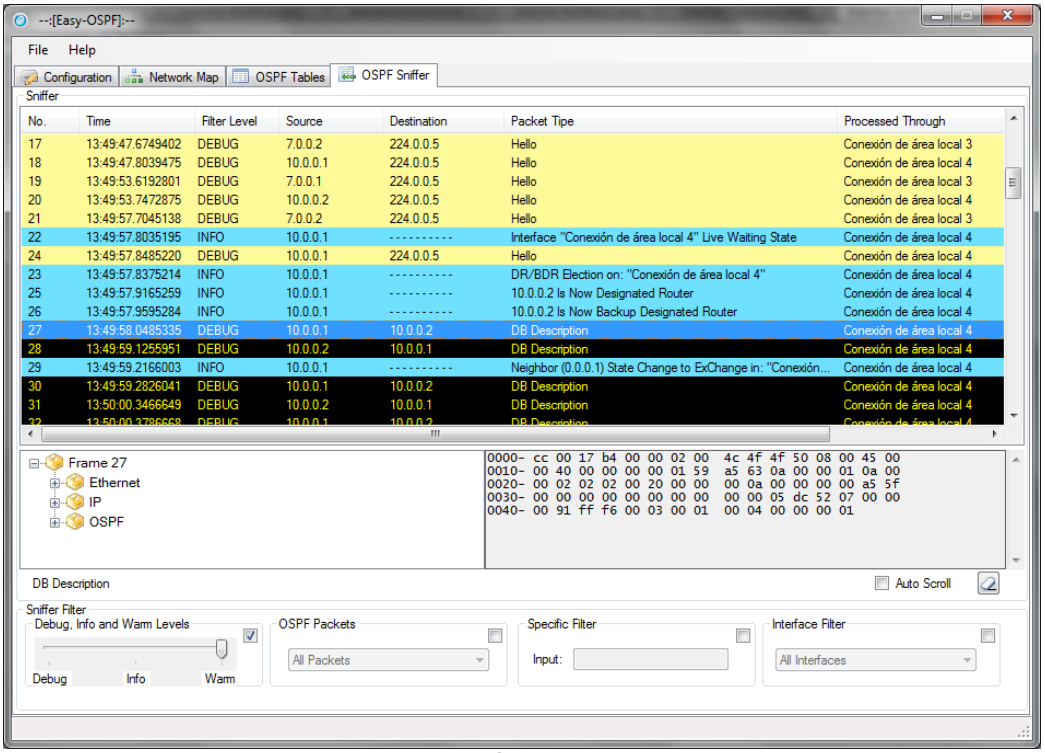

Figura 5.11: Módulo OSPF Sniffer

5.7.4 Fase de Pruebas

Para las pruebas que se realizaron para comprobar el correcto funcionamiento de este módulo se procedieron a ejecutar ensayos de sincronización de bases de datos topológicas, descubrimiento de vecinos, entre otras, y así comparar los resultados mostrados con otras aplicaciones que cumplen con el mismo fin, como Wireshark[2].

5.8 Fase de Pruebas y Depuración Final

Luego de terminar la fase de desarrollo de Easy-OSPF se construyeron diferentes escenarios, en los cuales la aplicación era probada profundamente para ubicar, y así poder arreglar todas aquellas fallas en cada uno de los módulos de Easy-OSPF. A continuación se especifican los escenarios antes mencionados:

5.8.1 Escenario 1

Escenario simple (Figura 5.12), consta de 2 routers conectados al PC donde está instalado Easy-OSPF. Este escenario permite desarrollar situaciones en las cuales se puede estudiar el comportamiento básico de la aplicación así como sus posibles fallas. Los puntos importantes probados en este escenario se especifican en la Tabla 5.1.

Figura 5.12: Topología Escenario 1

Descripción de la Prueba	Módulos Involucrados	Fallas Observadas en los Respectivos Módulos
Configuración de las dos interfaces a utilizar del PC, incluyendo cambio de direcciones IP y máscaras.	• Configuration • Network Map • OSPF Tables • OSPF Sniffer	**Configuration:** Alerta de solapamiento de segmentos de red no implementada.
Inicialización de procesos OSPF en ambas interfaces.	• Configuration • Network Map • OSPF Tables • OSPF Sniffer	**OSPF Sniffer:** Fallas en la selección del paquete capturado. **OSPF Tables**: Generación correcta del Router LSA, de la aplicación.
Pérdida de un vecino.	• Network Map • OSPF Tables • OSPF Sniffer	**Network Map:** No eliminaba al vecino de la gráfica.
Descubrimiento de un nuevo vecino.	• Network Map • OSPF Tables • OSPF Sniffer	
Pérdida de una interfaz OSPF en la aplicación.	• Configuration • Network Map • OSPF Tables • OSPF Sniffer	No realiza el flooding de la actualización de la Base de Datos Topológica.

Tabla 5.1: Escenario 1

Para darle solución a las fallas encontradas se implementaron una serie de métodos que se mencionan a continuación:

- Se implementó un método de validación para las direcciones IP de las interfaces de red, que verifique el formato de las mismas así como la existencia de solapamiento entre ellas.
- Se revisó la generación del router LSA de la aplicación corrigiendo los campos concernientes a la descripción de las interfaces del router.
- Se realizaron validaciones en el módulo de graficación en cuanto a la eliminación de routers no vecinos en el sistema autónomo.
- Se implementaron las reglas concernientes a la escogencia de interfaces por la cual se realizara el *flooding reliable*.

5.8.2 Escenario 2

En el escenario 2 se tiene la misma topología del escenario 1 con la diferencia que el PC en el cual está corriendo Easy-OSPF tiene dos interfaces Loopback OSPF (Figura 5.13). Los puntos importantes probados en este escenario (Tabla 5.2).

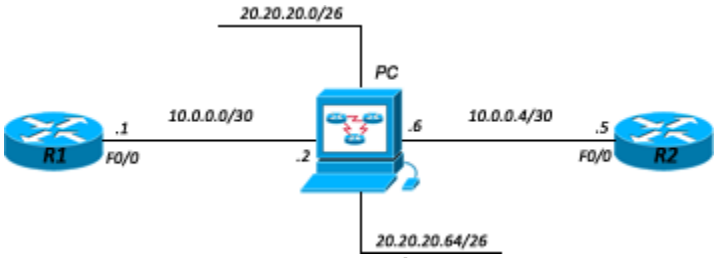

Figura 5.13: Topología Escenario 2

Descripción de la Prueba	Módulos Involucrados	Fallas Observadas en los Respectivos Módulos
Configuración de las dos interfaces a utilizar del PC, incluyendo cambio de direcciones IP y máscaras.	• Configuration • Network Map • OSPF Tables • OSPF Sniffer	Problemas en el cambio de IP de la interfaz en Windows 7.
Inicialización de procesos OSPF en las interfaces.	• Configuration • Network Map • OSPF Tables • OSPF Sniffer	
Pérdida de un vecino.	• Configuration • Network Map • OSPF Tables • OSPF Sniffer	No realizaba el *flooding reliable* de manera exitosa.
Descubrimiento de un nuevo vecino.	• Configuration • Network Map • OSPF Tables • OSPF Sniffer	

Pérdida de una interfaz OSPF en la aplicación.	• Configuration • Network Map • OSPF Tables • OSPF Sniffer	

Tabla 5.2: Escenario 2

Para la solución de las fallas presentadas en este escenario se tomaron las siguientes medidas:

- Se investigó, y posteriormente implementó, la forma con la cual Windows realiza el cambio de IP de una interfaz de red, ya que en las versiones de dicho sistema operativo hasta XP se realiza con un formato específico de instrucciones, las cuales cambian para Windows Vista y 7.
- Se revisaron y modificaron la implementación de las reglas para la realización del flooding reliable en las respectivas interfaces.

5.8.3 Escenario 3

Este escenario consta de 3 routers (R1, R2 y R3) conectados en el mismo segmento del PC con Easy-OSPF a través de un switch. Cada uno de estos router posee una interfaz Loopback (Figura 5.14). Dicho escenario se definió especialmente para probar la elección del DR y BDR, así como el flooding de los cambios en la topología. Los puntos importantes probados en este escenario se especifican en la Tabla 5.3.

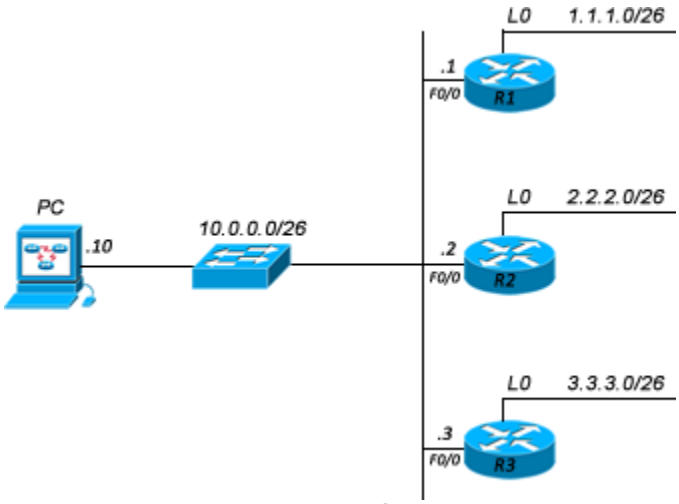

Figura 5.14: Topología Escenario 3

[2] http://www.wireshark.org

Descripción de la Prueba	Módulos Involucrados	Fallas Observadas en los Respectivos Módulos
Configuración de la interfaz a utilizar del PC, incluyendo cambio de dirección IP y máscara.	• Configuration • Network Map • OSPF Tables • OSPF Sniffer	
Inicialización del proceso OSPF en la interfaz física.	• Configuration • Network Map • OSPF Tables • OSPF Sniffer	Sincronizaba con todos los vecinos sin ser la aplicación DR o BDR.
Levantamiento de Interfaz Loopback en un vecino.	• Network Map • OSPF Tables • OSPF Sniffer	**OSPF Tables:** No se recalculaba la tabla de enrutamiento.
Pérdida de Interfaz Loopback en un vecino.	• Network Map • OSPF Tables • OSPF Sniffer	**Network Map:** No graficaba la pérdida de la interfaz correctamente.
Pérdida de un vecino.	• Network Map • OSPF Tables • OSPF Sniffer	
Descubrimiento de un nuevo vecino.	• Network Map • OSPF Tables • OSPF Sniffer	
Pérdida de una interfaz OSPF en la aplicación.	• Network Map • OSPF Tables • OSPF Sniffer	Al pasar al modo “Passive” seguían enviando mensaje Hello a través de ella. Además no tomaba en cuenta esta interfaz al construir el Router LSA de la aplicación.

Tabla 5.3: Escenario 3

Debido a las fallas registradas en el escenario anteriormente descrito se implementaron las siguientes soluciones:

- El protocolo OSPF especifica que los todos los routers en un área deben sincronizar únicamente sus bases de datos topológicas con DR y el BDR, por ende se debe implementar que entre otro tipo de router no ocurra esta sincronización.
- Se modificó el momento en el cual se recalculaba el SPT.
- Se realizaron nuevas validaciones para soportar la perdida de interfaces que no pertenecieran al router.
- Se implemento código separado para cada uno de los posibles estados de la interfaz (*enable, passive, disable*) para realizar las acciones concernientes a cada estado de manera sencilla.

5.8.4 Escenario 4

Ya que OSPF es un protocolo en el cual cada router tiene un conocimiento completo de la topología de red, se desarrollaron escenarios, como el que se puede apreciar en la Figura 5.15. Para esto se definieron 2 routers (R1 y R2), además del PC con Easy-OSPF, conectado linealmente de manera que R2 no actuara como un vecino de la aplicación. Los puntos importantes probados en este escenario se especifican en la Tabla 5.4.

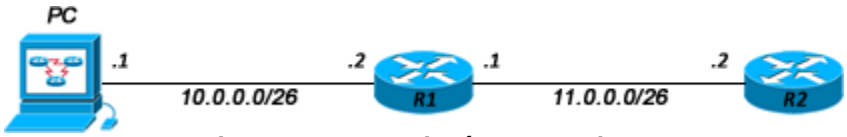

Figura 5.15: Topología Escenario 4

Descripción de la Prueba	Módulos Involucrados	Fallas Observadas en los Respectivos Módulos
Configuración de la interfaz a utilizar del PC, incluyendo cambio de dirección IP y máscara.	• Configuration • Network Map • OSPF Tables • OSPF Sniffer	
Inicialización del proceso OSPF en la interfaz física.	• Configuration • Network Map • OSPF Tables • OSPF Sniffer	
Conocimiento de un nuevo router (R2).	• Configuration • OSPF Tables • OSPF Sniffer	
Perdida del router R2.	• Network Map • OSPF Tables • OSPF Sniffer	**Network Map:** La gráfica de la topología no se actualizaba. **OSPF Tables:** No se recalculaba la tabla de enrutamiento.
Pérdida de un vecino.	• Network Map • OSPF Tables • OSPF Sniffer	
Descubrimiento de un nuevo vecino.	• Network Map • OSPF Tables • OSPF Sniffer	
Pérdida de una interfaz OSPF en la aplicación.	• Network Map • OSPF Tables • OSPF Sniffer	
Cambios entre los modos de interfaz OSPF (enable, passive y disable).	• Configuration • Network Map • OSPF Tables • OSPF Sniffer	

Tabla 5.4: Escenario 4

Al igual que en anteriores escenarios se encontraron fallas en la aplicación, debido a las cuales se implementaron las siguientes soluciones:

- Desarrollo de validaciones en cuanto a la información contenida en la base de datos topológica para la graficación correcta de la topología.
- Se revisó y modificó el algoritmo de Dijkstra utilizado en el SPT para adaptarlo a la aplicación.

5.8.5 Escenario 5

Al igual que el escenario anterior, dentro de las distintas pruebas hechas se pretende revisar el comportamiento del módulo "Network Map" al graficar la topología de manera completa (Figura 5.16). Los puntos importantes probados en este escenario se especifican en la Tabla 5.5.

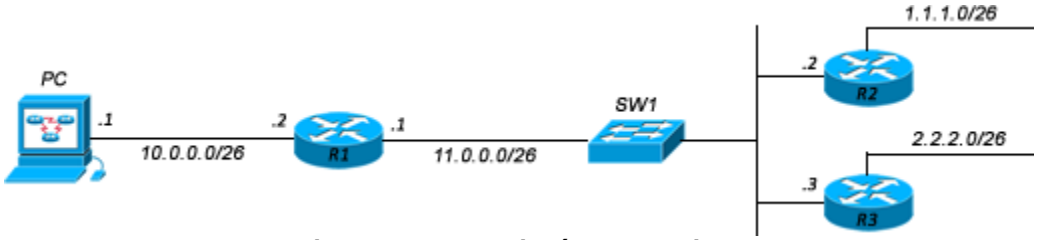

Figura 5.16: Topología Escenario 5

Descripción de la Prueba	Módulos Involucrados	Fallas Observadas en los Respectivos Módulos
Configuración de la interfaz a utilizar del PC, incluyendo cambio de dirección IP y máscara.	• Configuration • Network Map • OSPF Tables • OSPF Sniffer	
Inicialización del proceso OSPF en la interfaz física.	• Configuration • Network Map • OSPF Tables • OSPF Sniffer	
Sincronización de la Base de Datos Topológica.	• OSPF Tables • OSPF Sniffer	
Descubrimiento de un nuevo router (R2 o R3).	• Configuration • OSPF Tables • OSPF Sniffer	
Pérdida del router R2 ó R3.	• Network Map • OSPF Tables • OSPF Sniffer	**Network Map:** La gráfica de la topología no se actualizaba. **OSPF Tables:** No se recalculaba la tabla de enrutamiento.
Pérdida de un vecino.	• Network Map • OSPF Tables • OSPF Sniffer	

Conocimiento de un nuevo vecino.	• Network Map • OSPF Tables • OSPF Sniffer	
Pérdida de una interfaz OSPF en la aplicación.	• Network Map • OSPF Tables • OSPF Sniffer	
Cambios entre los modos de interfaz OSPF (enable, passive y disable).	• Configuration • Network Map • OSPF Tables • OSPF Sniffer	

Tabla 5.5: Escenario 5

Como en escenarios anteriores Easy-OSPF presentó debido a las cuales se implementaron las siguientes soluciones:

- Desarrollo de validaciones en cuanto a la información contenida en la base de datos topológica para la graficación correcta de la topología, al validar el no tomar en cuenta información contenida en dicha base de datos perteneciente a routers no activos.
- Se revisó y modificó el algoritmo de Dijkstra utilizado en el SPT para adaptarlo a la aplicación.

5.8.6 Escenario 6

En el último de los escenarios desarrollados, se dispuso de una serie de routers, colocados como se puede apreciar en la Figura 5.17, para realizar pruebas más cercanas a la realidad y en la cuales se evaluaría Easy-OSPF en una serie de puntos referentes al protocolo. Los puntos importantes probados en este escenario se especifican en la Tabla 5.6.

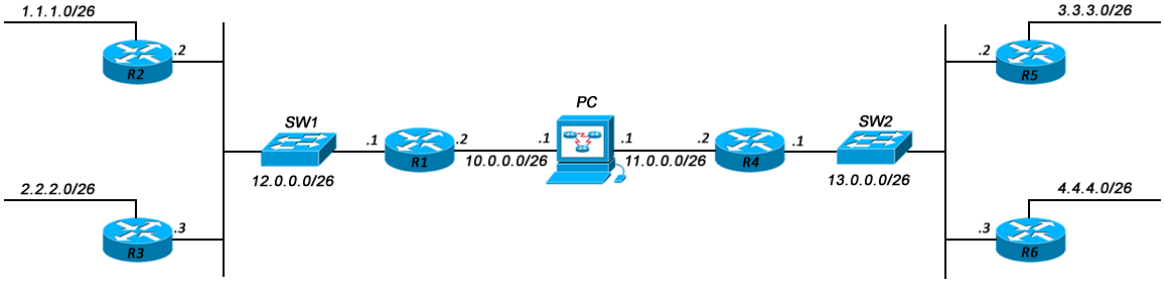

Figura 5.17: Topología Escenario 6

Descripción de la Prueba	Módulos Involucrados	Fallas Observadas en los Respectivos Módulos
Configuración de la interfaz a utilizar del PC, incluyendo cambio de dirección IP y máscara.	• Configuration • Network Map • OSPF Tables • OSPF Sniffer	
Inicialización del proceso OSPF en la interfaz física.	• Configuration • Network Map • OSPF Tables • OSPF Sniffer	
Sincronización de la Base de Datos Topológica.	• OSPF Tables • OSPF Sniffer	
Conocimiento de un nuevo router (R2 o R3).	• Configuration • OSPF Tables • OSPF Sniffer	
Pérdida del router R2 ó R3.	• Network Map • OSPF Tables • OSPF Sniffer	**Network Map:** La gráfica de la topología no se actualizaba. **OSPF Tables:** No se recalculaba la tabla de enrutamiento.
Pérdida de un vecino.	• Network Map • OSPF Tables • OSPF Sniffer	
Conocimiento de un nuevo vecino.	• Network Map • OSPF Tables • OSPF Sniffer	
Pérdida de una interfaz OSPF en la aplicación.	• Network Map • OSPF Tables • OSPF Sniffer	No realizaba el flooding correctamente entre interfaces de Easy-OSPF.
Cambios entre los modos de interfaz OSPF (enable, passive y disable).	• Configuration • Network Map • OSPF Tables • OSPF Sniffer	

Tabla 5.6: Escenario 6

El escenario anteriormente descrito surgiendo algunas fallas descritas en la Tabla 6.6 por lo cual se implementaron las siguientes soluciones:

- Implementación de validaciones concernientes a la información de la base de datos topológica para la graficación de la topología de red.
- Revisión y corrección de las políticas para escoger las interfaces por las cuales se realiza el flooding reliable.

5.9 Especificaciones Técnicas de Easy-OSPF

Easy-OSPF fue desarrollado utilizando computadores con el sistema operativo Windows (XP, Vista, y 7) y con Visual Studio 2008 como entorno de desarrollo.

Todas las pruebas, en la fase de desarrollo y depuración, se hicieron utilizando los routers Cisco situados en el Laboratorio de Internet II, ubicado en la Facultad de Ciencias de la Universidad Central de Venezuela, en conjunto al simulador de redes GNS3 utilizando la imagen del Cisco IOS (*Internetwork Operating System*) *c3640-ik9o3s-mz.124-7.*

Con respecto a las librerías utilizadas, Easy-OSPF fue desarrollada y probada con las siguientes versiones: *PacketDotNet* v0.4, *SharpPcap* v3.1, *Log4Net* v1.2.9, y *Piccolo2D* 1.0.9.

Easy-OSPF debe ser ejecutado en modo administrador, de lo cual se encarga el UAC para Windows Vista y Windows 7.

6 Sondeo de Exploración y Resultados

Luego de concluida la depuración de la aplicación se procedió a realizar un sondeo de investigación entre una población de 20 alumnos, pertenecientes a diversos cursos de redes, de la Escuela de Computación de la Universidad Central de Venezuela con el fin de medir la funcionalidad de Easy-OSPF por parte de los usuarios. Se realizo en el Laboratorio de Internet II ubicado en el Galpón 10 de la Escuela de Computación de la Universidad Central de Venezuela y se utilizó GNS3 junto a la aplicación en diferentes topologías determinadas a fin de probar los diferentes controles de Easy-OSPF.

Luego de las pruebas se le proporcionó una encuesta con una serie de preguntas sobre Easy-OSPF las cuales fueron divididas en 8 secciones tal como se puede apreciar en el modelo de encuesta que se encuentra en el Apéndice A.

El sondeo utilizó una puntuación del 1 al 5, siendo 1 la peor calificación y 5 la mayor. De esto se desprendieron una serie de resultados los cuales se plasmaron en una serie de gráficos a continuación presentados. Estos gráficos se muestran por porcentajes de cada calificación por pregunta realizada en la encuesta.

6.1 Conocimientos Generales

En esta sección del sondeo se pregunta sobre los conocimientos generales del estudiante en el área de redes, protocolos de enrutamiento y específicamente OSPF. Estas preguntas pueden verse en el modelo de encuesta presentado en el Apéndice A. Los resultados de esta sección se presentan en la Figura 6.1.

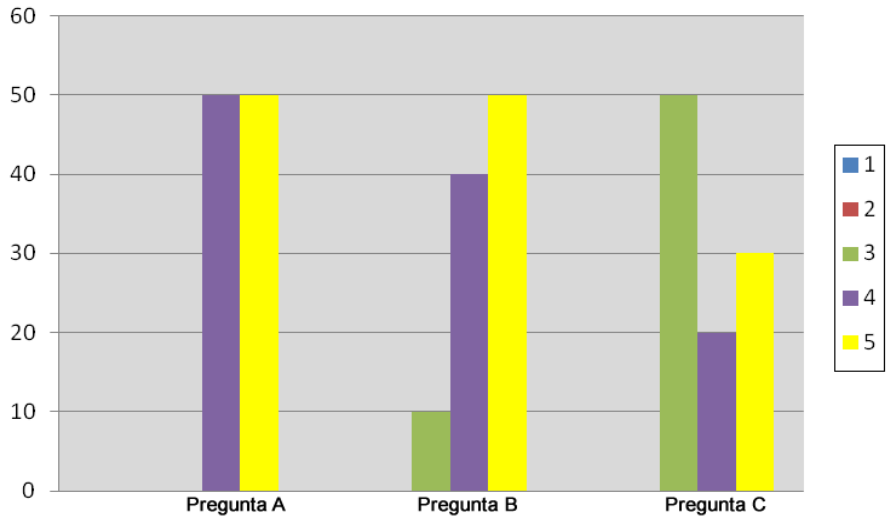

Figura 6.1: Valoración de Conocimientos Generales

6.2 Valoración de la Aplicación en Forma General

En esta sección de la encuesta se trata de medir el aspecto general de Easy-OSPF en cuanto a usabilidad, iconos, metáforas, navegación e información mostrada. Las preguntas pueden verse en el modelo de encuesta del Apéndice A. Los resultados se plasmaron en la Figura 6.2 y la Figura 6.3.

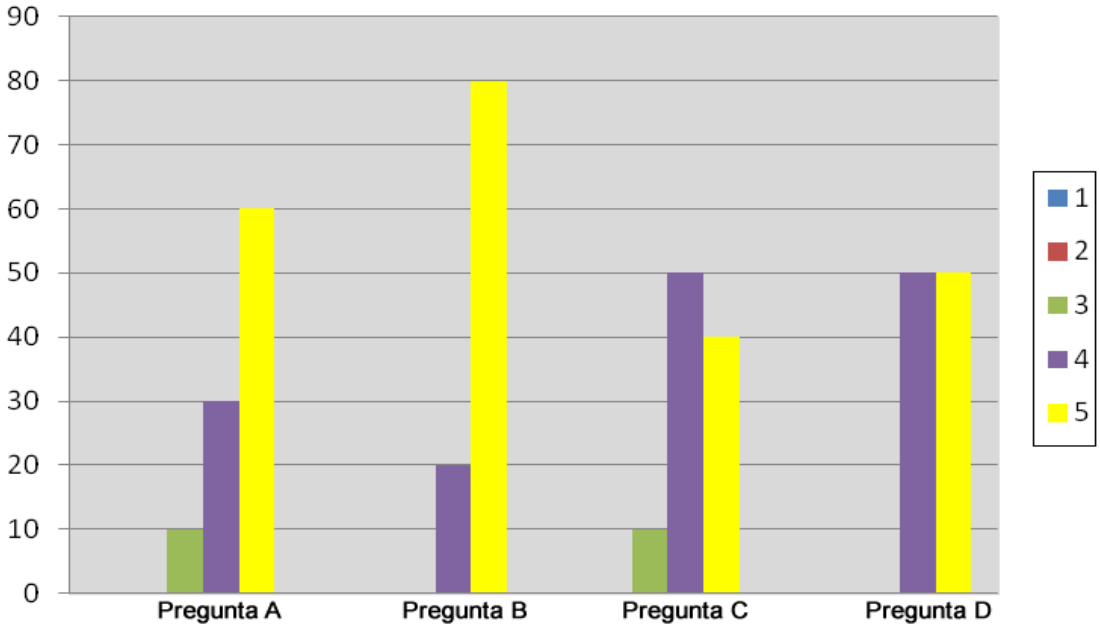

Figura 6.2: Valoración General de la Aplicación (Preguntas A - D)

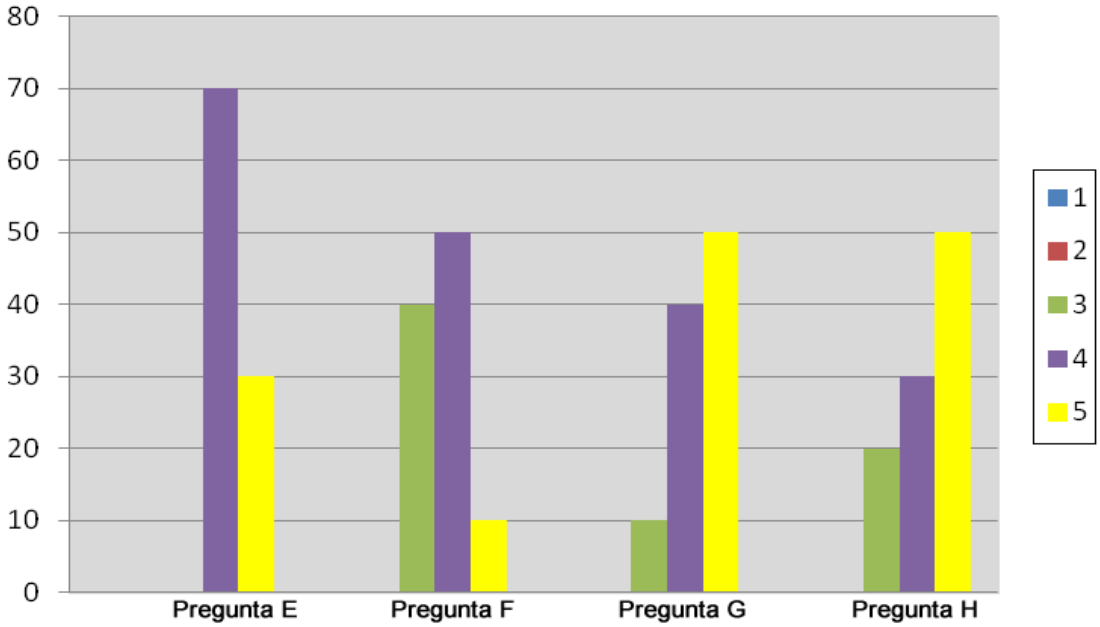

Figura 6.3: Valoración General de la Aplicación (Preguntas E - H)

6.3 Valoración del Módulo Configuration

Las preguntas de esta sección van enfocadas específicamente en el módulo Configuration de Easy-OSPF, sus funcionalidades, mensajes y su aspecto. La preguntas de la referida sección pueden verse en el modelo de encuesta del Apéndice A y los resultados están expresados de manera gráfica en la Figura 6.4.

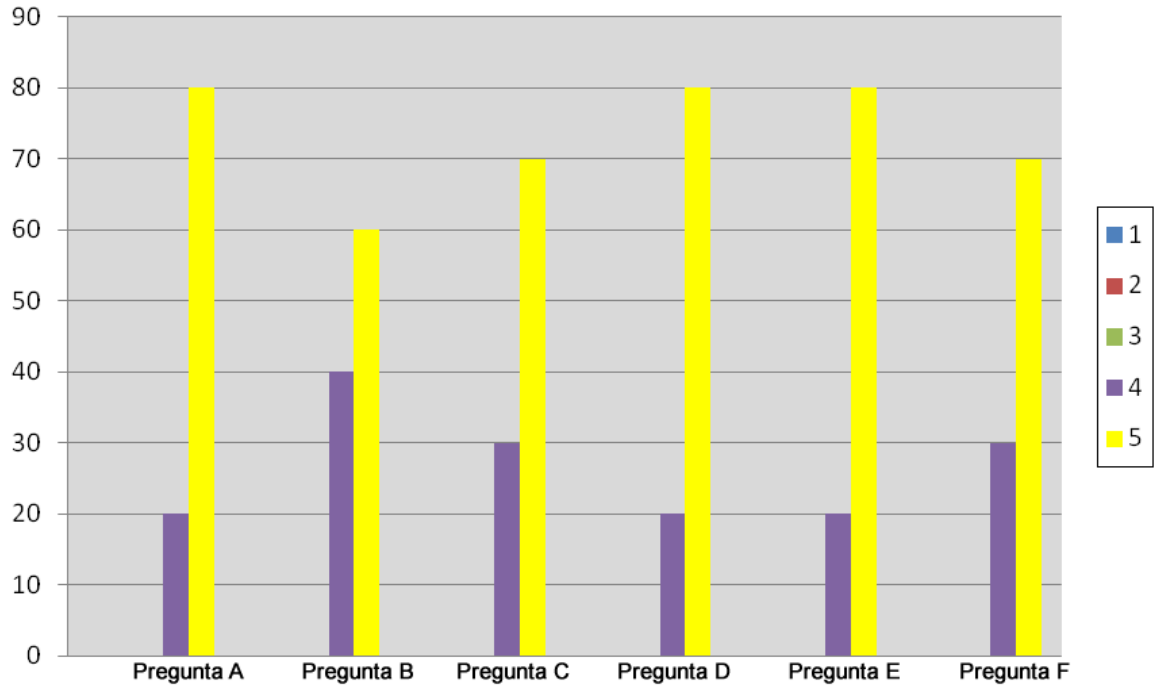

Figura 6.4: Valoración del Módulo Configuration

6.4 Valoración del Módulo Network Map

Las preguntas de esta sección van enfocadas específicamente en el módulo Network Map de Easy-OSPF, la forma de graficar la topología y los controles para manejar dicha imagen. Los resultados son presentados en la Figura 6.5.

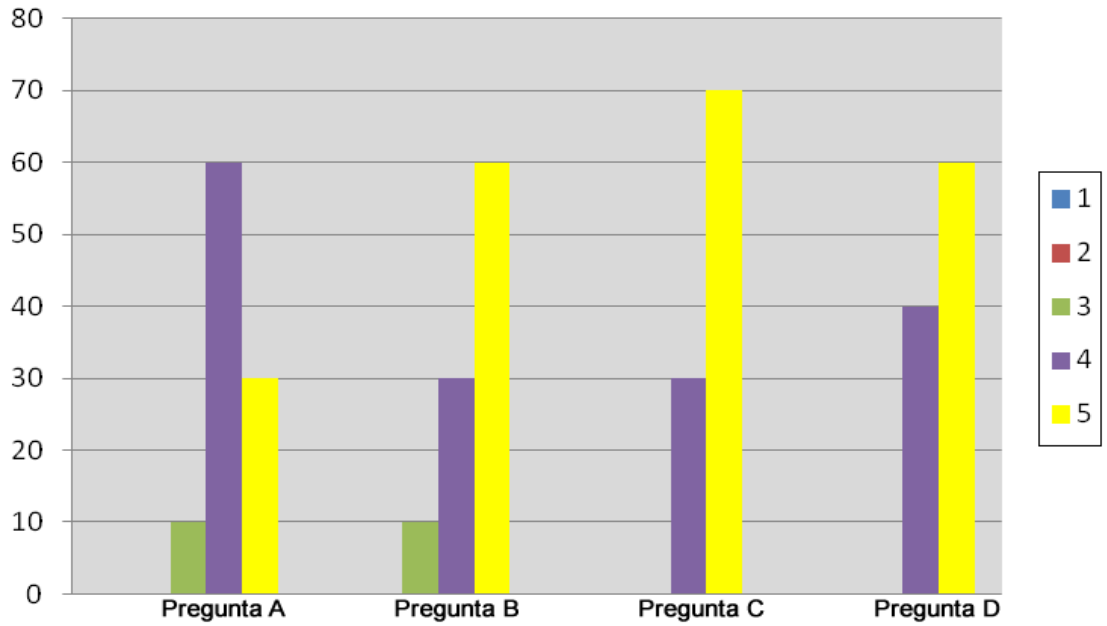

Figura 6.5: Valoración del Módulo Network Map

6.5 Valoración del Módulo OSPF Tables

Aquí se enfocó el sondeo en investigar si la forma de mostrar la información de las diversas tablas involucradas en OSPF está implementada de manera legible para el usuario (Figura 6.6).

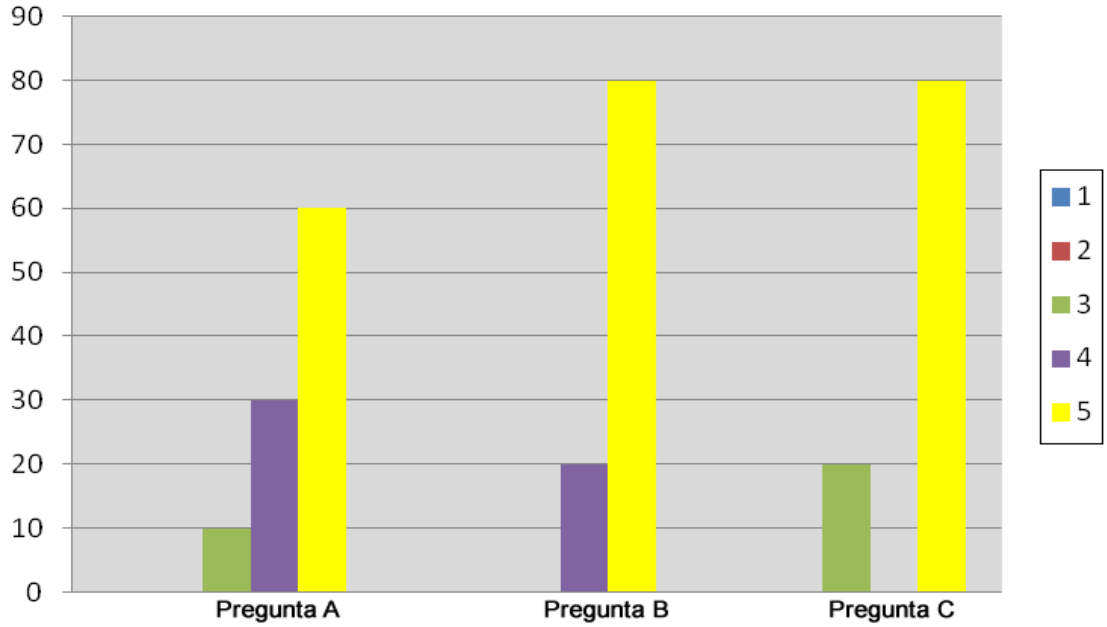

Figura 6.6: Valoración del Módulo OSPF Tables

6.6 Valoración del Módulo OSPF Sniffer

Igual que los otros módulos es importante definir si la información, y la forma como esta es mostrada, es fácil de captar por el usuario y no le ocasionan problemas para estudiar el estado del protocolo en cierto momento. Las preguntas para definir esto se pueden apreciar en el Apéndice A. Los resultados están expresados en la Figura 6.7.

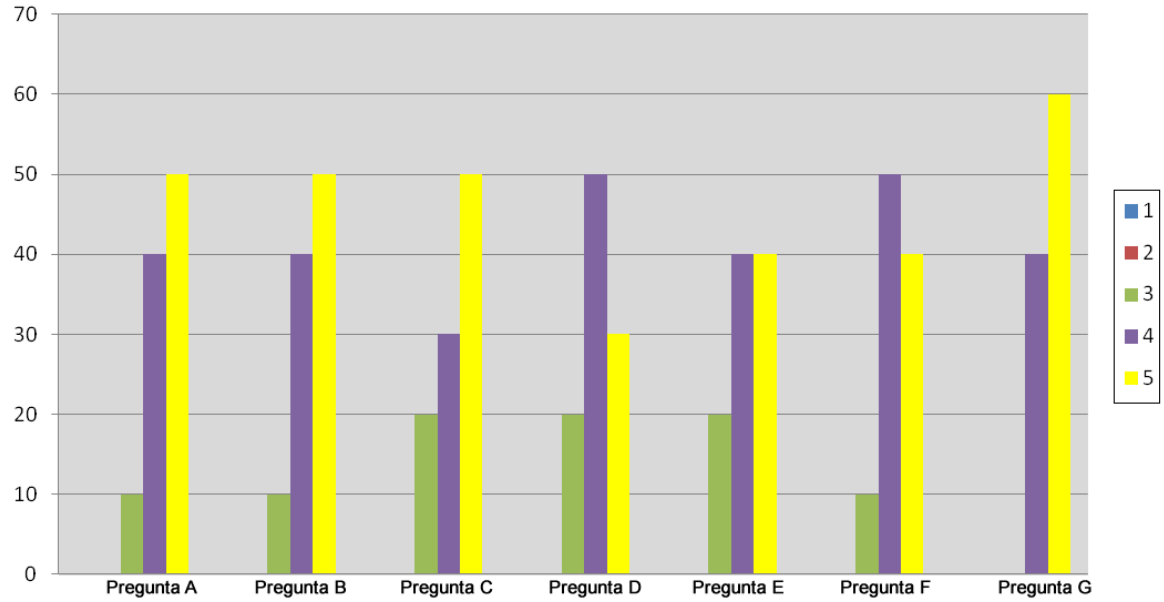

Figura 6.7: Valoración del Módulo OSPF Sniffer

6.7 Consideraciones Finales

Al final de la encuesta se definieron ciertas preguntas para determinar si la aplicación satisface su objetivo didáctico y facilita el aprendizaje de OSPF en comparación a otras herramientas disponibles. Estas preguntas pueden verse en el modelo de encuesta que se encuentra en el Apéndice A y los resultados se encuentran plasmados en la Figura 6.8.

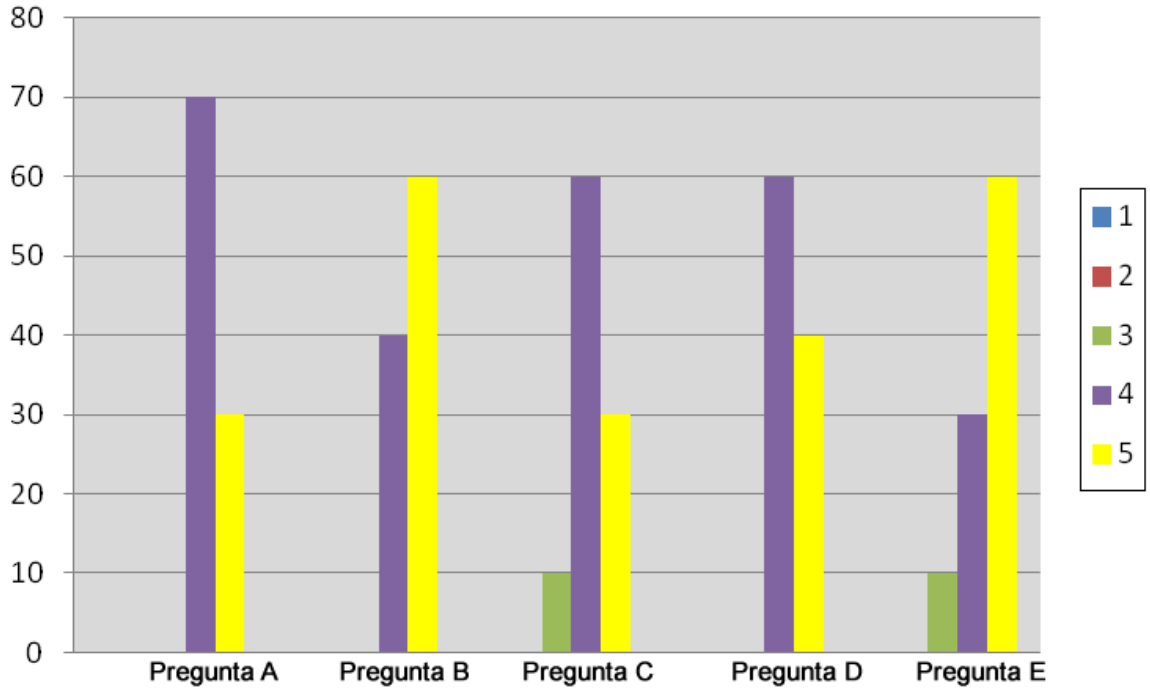

Figura 6.8: Valoraciones de Consideraciones Finales

Además se determino, con ciertas preguntas de la encuesta, algunos errores de la aplicación, los cuales fueron depurados posteriormente. De igual forma algunos usuarios realizaron comentarios, que a su parecer, pueden mejorar la aplicación, estas apreciaciones fueron tomadas en cuenta al terminar de depurar la aplicación.

También se puede desprender de este estudio el hecho de que Easy-OSPF es vista como una herramienta fácil de usar y entender lo cual refuerza su condición de herramienta didáctica.

7 Conclusiones

Easy-OSPF nace con el fin de proveer una herramienta didáctica que mediante un conjunto de recursos y herramientas facilite el proceso de enseñanza–aprendizaje entre profesores y alumnos, permitiendo la interactividad y retroalimentación de lo aprendido.

Basándose en lo anterior como principal premisa se proporcionó a la aplicación, Easy-OSPF, una serie de características que satisfacen con éxito los objetivos planteados al comienzo del desarrollo del presente Trabajo Especial de Grado:

- Interfaz grafica intuitiva y fácil de manejar.
- Módulos con tareas específicas que detallan las partes más importantes del funcionamiento del protocolo.
- Herramientas para manejar la graficación de la topología.
- Herramientas para el filtrado de la información mostrada.
- Exposición de la información de forma clara para el usuario.
- Almacenamiento de configuraciones para su posterior estudio.
- Implementación de un modulo de manejo de logs para eventos específicos del protocolo.

En comparación a otras implementaciones de OSPF, Easy-OSPF da un paso más allá, puesto que se enfoca especialmente en la parte didáctica, mostrándole al usuario la mayor cantidad de información posible para la comprensión del protocolo, convirtiéndose así en una herramienta útil tanto para el instructor como para el estudiante. Esto se demostró con los resultados del sondeo de exploración realizado y que se detalla en el Capítulo 6.

Easy-OSPF muestra de manera detalla la información contenida en la base de datos topológica, tabla de vecinos y de enrutamiento, las cuales reflejan el resultado de las acciones del protocolo. También incluye un sniffer dedicado de paquetes OSPF, lo cual significa un ahorro en el número de aplicaciones necesarias para el estudio del intercambio de información que se da en el protocolo, las cuales Easy-OSPF integra. Todo esto le permite al usuario un mayor enfoque al analizar el protocolo.

8 Recomendaciones y Trabajos Futuros

Aunque Easy-OSPF alcanzó con éxito los objetivos principales propuestos al inicio existen una serie de módulos adicionales, o extensiones de los ya existentes, que pueden desarrollarse para la complementación de la aplicación. Podemos mencionar:

- Expansión del protocolo a múltiples áreas. Ya que Easy-OSPF es una implementación a nivel se single-area.
- Modulo de graficación del SPT (Shortest Path Tree). Puesto que en el proceso de estudio de OSPF se verificó que esta estructura, que guarda las mejores rutas, es de vital importancia para llevar a cabo el fin de este protocolo.
- Implementación de la autenticación MD5 del protocolo.

De los resultados obtenidos se abren sin lugar a dudas un conjunto amplio de posibles líneas de investigación, entre ellas citaremos las que nos parecen de mayor interés y relación con el presente trabajo:

- Desarrollo de herramientas didácticas para otros protocolos de enrutamiento existentes (RIP, RIPv2, IS-IS).
- Creación de una pila de aplicaciones didácticas orientadas a cursos de redes.

Referencias Bibliográficas

[1] W. Parkhurst. Cisco Router OSPF. McGraw-Hill. Julio, 1998.

[2] T. Thomas. OSPF Network Design Solutions, Second Edition. Cisco Press. Abril, 2003.

[3] J. Doyle. Routing TCP/IP. Volume I. Octubre, 2005.

[4] W. Goralski. Juniper and Cisco Routing, Policy and Protocols for Multivendor Networks. First Edition. Wiley Publishing Inc. Septiembre, 2002.

[5] D. Medhi and K. Ramasamy. Network Routing, Algorithms, Protocols and Architectures. First Edition. Morgan Kaufmann Publishers. Abril, 2007.

[6] S. Halabi. OSPF Design Guide. Revision 1.1. NSA Group. Abril, 1996.

[7] J. Moy. OSPF, Anatomy of an Internet Routing Protocol. First Edition. Addison-Wesley. Febrero, 1998.

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

[9] G. Malkin. RIP Versión 2. RFC 2453. Noviembre, 1998.

[10] C. Hendrick. RIP. RFC 1058. Junio, 1998.

[11] R. Bellman, Dynamic Programming. Princeton University Press. Agosto, 1957.

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

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

[14] J. Postel. User Datagram Protocol. RFC 768. Agosto, 1980.

[15] Cisco System. “EIGRP”. White Papers.

[16] D. Fishburne. EIGRP Deployment. Cisco Networkers. First Edition. Mayo, 2006.

[17] A. Farrell. The Internet and its Protocols. First Edition. Morgan Kauffman. Septiembre, 2004.

[18] J. Doyle. OSPF and IS-IS: Choosing an IGP for Large Scale Networks. First Edition. Addison-Wesley Professional. Noviembre, 2005.

[19] W. Simpson. The Point-to-Point Protocol. RFC 1661. Julio, 1994.

[20] V. S. Bagad. Computer Networks I. Second Edition. Technical Publications Pune. Junio, 2006.

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

[22] ISO/IEC 23270. C# Language Especification. 2003.

[23] J. Exposito, T. Valentina and E. Gamess. Easy-EIGRP: A Didactic Application for Teaching and Learning of the Enhanced Interior Gateway Routing Protocol. 10.1109/ICNS.2010.53. Marzo 2010.

[24] J. Exposito, T. Valentina and E. Gamess. Implementación de un Protocolo de Enrutamiento Avanzado de Vector de Distancia. UCV. Enero 2010.

Apéndice A

Encuesta Easy-OSPF

Coloque un número del 1 al 5, donde 1 representa el menor puntaje y 5 el mayor

1. Conocimientos Generales

A. ¿Posee usted conocimientos sobre redes de computadores?

1 2 3 4 5

B. ¿Posee usted conocimientos sobre protocolos de enrutamiento?

1 2 3 4 5

C. ¿Tienes conocimientos generales sobre el funcionamiento de OSPF?

1 2 3 4 5

2. Valoración de la Aplicación en Forma General

A. ¿Cómo califica la usabilidad de la aplicación?

1 2 3 4 5

B. ¿Cómo califica la navegabilidad entre los módulos de la aplicación?

1 2 3 4 5

C. ¿Cómo califica la información de la barra de menús?

1 2 3 4 5

D. ¿Los iconos son suficientemente explicativos?

1 2 3 4 5

E. ¿Los textos de ayuda (tooltips) son claros y proveen la información deseada?

1 2 3 4 5

F. ¿La informacion de los mensajes de error es clara y concreta?

1	2	3	4	5

G. ¿Las imágenes y controles proporcionan la información correcta del estado de la aplicación?

1	2	3	4	5

H. ¿Los controles de la aplicación son explicativos en cuanto a su función?

1	2	3	4	5

3. Módulo Configuration

A. ¿Cómo califica el manejo de direcciones lP?

1	2	3	4	5

B. ¿Cómo califica la capacidad de Easy-OSPF para detectar cambios externos de las interfaces del PC?

1	2	3	4	5

C. ¿Cómo califica la facilidad de configuración para los campos Metric, Hello Interval y Dead Interval?

1	2	3	4	5

D. ¿Cómo califica la utilidad de la imagen del estatus de la interfaz?

1	2	3	4	5

E. ¿Cómo califica la navegabilidad entre las distintas interfaces descubiertas por Easy-OSPF?

1	2	3	4	5

F. ¿Cómo califica el orden de la interfaz, es agradable a la vista?

1	2	3	4	5

4. Módulo Network Map

A. ¿Cómo califica el proceso de graficación de la topología?

1	2	3	4	5

B. ¿Cómo califica la imágen de la Topología (tamaño, visibilidad, texto, etc)?

1	2	3	4	5

C. ¿Cómo califica el manejo de los controles de imagen (Zoom, Save Topology, Drag/Piking Mode)?

1	2	3	4	5

D. ¿Cómo califica el orden de la interfaz, es agradable a la vista?

1	2	3	4	5

5. Módulo OSPF Tables

A. ¿Cómo califica la presentación de la información?

1	2	3	4	5

B. ¿Cómo califica el orden de la interfaz, es agradable a la vista?

1	2	3	4	5

C. ¿Cómo califica la selección de los LSA en la base de datos topológica?

1	2	3	4	5

D. ¿Qué agregaría o quitaría de este módulo y por qué?

_____

_____

_____

6. Módulo OSPF Sniffer

A. ¿Los campos en el sniffer son suficientes y explicativos?

1	2	3	4	5

B. ¿Cómo califica la forma de colorear los mensajes?

1	2	3	4	5

C. ¿Cómo califica la manera de deplegar la información luego de seleccionar una fila del sniffer?

1	2	3	4	5

D. ¿Cómo califica la presentación en bytes de los paquetes OSPF?

1	2	3	4	5

E. ¿Cómo califica la utilidad de los filtros del sniffer?

1	2	3	4	5

F. ¿Es suficiente la información desplegada con respecto a las acciones tomadas por el usuario a través de Easy-OSPF?

1	2	3	4	5

G. ¿Es suficiente la información desplegada con respecto a las acciones sucedidas en la topología (detección vecinos, adyacencias)?

1	2	3	4	5

7. Consideraciones Finales.

A. ¿Qué tan fácil es aprender OSPF con esta herramienta?

1	2	3	4	5

B. ¿Qué tan fácil resulta usar la aplicación?

1	2	3	4	5

C. ¿Qué tan fácil resulta configurar un router?

1	2	3	4	5

D. ¿Qué tan fácil resulta configurar OSPF en un router?

1	2	3	4	5

E. ¿Qué tan fácil resulta observar los cambios configurados?

1	2	3	4	5

8. Comentarios Finales

A. De haber encontrado un error en la aplicación especifique:

_____

_____

_____

_____

B. ¿Qué agregaría o quitaría de la aplicación y por qué?:

_____

_____

_____

_____