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

wradvs: Un Servidor de Autoconfiguración sin Estado para IPv6 en Plataforma Windows

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

Gabriel Abadi Eskenazi
C.I.: 17.125.575
E-mail: gabriel.abadi@gmail.com

Francisco A. Jiménez Fuentes
C.I.: 18.247.608
E-mail: jimenezff@gmail.com

para optar al título de Licenciado en Computación

Tutor: Prof. Eric Gamess

Caracas, Junio 2009

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

ACTA DEL VEREDICTO

Quienes suscriben, Miembros del Jurado designado por el Consejo de la Escuela de Computación para examinar el Trabajo Especial de Grado, presentado por los Bachilleres Gabriel Abadi C.I.: 17.125.575 y Francisco Jiménez C.I.: 18.247.608, con el título **"wradvs: Un Servidor de Autoconfiguración sin Estado para IPv6 en Plataforma Windows"**, a los fines de cumplir con el requisito legal para optar al título de Licenciado en Computación, dejan constancia de lo siguiente:

Leído el trabajo por cada uno de los Miembros del Jurado, se fijó el día <día> de <mes> de <año>, a las <horaPresentación>, para que sus autores lo defendieran en forma pública, en <lugarPresentación>, lo cual estos realizaron mediante una exposición oral de su contenido y luego respondieron satisfactoriamente a las preguntas que les fueron formuladas por el Jurado, todo ello conforme a lo dispuesto en la Ley de Universidades y demás normativas vigentes de la Universidad Central de Venezuela. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo.

En fe de lo cual se levanta la presente acta, en Caracas el <día> de <mes> de <año>, dejándose también constancia de que actuó como Coordinador del Jurado el Profesor Tutor Eric Gamess.

_____
Prof. Eric Gamess
(Tutor)

_____
Prof. Juan Carlos Fernández
(Jurado Principal)

_____
Prof. Robinson Rivas
(Jurado Principal)

RESUMEN

TÍTULO
wradvs: Un Servidor de Autoconfiguración sin Estado para IPv6 en Plataforma Windows.

AUTORES
Gabriel Abadi E.
Francisco A. Jiménez F.

TUTOR
Prof. Eric Gamess

El presente Trabajo Especial de Grado consiste en el desarrollo de una aplicación denominada “wradvs” que lleva a cabo el proceso de autoconfiguración sin estado de IPv6 a través de un servicio de Windows.

El documento comienza con un estudio teórico de todos los conceptos necesarios para la comprensión del proceso de autoconfiguración sin estado incluyendo los RFC 2460, 4861, 5006, 4191, 3775 y 3963. La implementación se lleva a cabo utilizando una metodología ágil de desarrollo denominada Scrum que propone ciclos cortos para la planificación y posterior codificación de los diferentes módulos de la aplicación.

La aplicación consta de cuatro módulos: el módulo de configuración del servidor permite almacenar todos los parámetros y opciones de los mensajes Router Advertisement en un archivo XML. El módulo principal es el servidor wradvs que lee este archivo para enviar mensajes Router Advertisement periódicos y en respuesta a mensajes Router Solicitation mientras llena un log de eventos con mensajes de información y error. Para la lectura del log se ofrece un visor de eventos que muestra la información en tiempo real. Por último se proporciona una herramienta para la administración de direcciones IPv6, rutas por defecto, tabla de enrutamiento y reenvío de paquetes. Cada módulo posee una amigable e intuitiva interfaz gráfica para sus usuarios.

El producto final junto a la documentación de la investigación está publicada como una aplicación con licencia GNU GPL a través de SourceForge.net (un sitio Web que permite el hospedaje y distribución de software libre y sistemas de código abierto) en el siguiente enlace: http://wradvs.sourceforge.net.

Palabras clave: stateless autoconfiguration, router advertisement, IPv6, Windows, service, Daemon, server.

Tabla de Contenido

Índice de Figuras

Índice de Tablas

Introducción

Con el crecimiento de las redes de computadores y de las telecomunicaciones, las direcciones IPv4 poco a poco se han ido agotando. Aún cuando este problema no afecta a nuestra sociedad actualmente, no hay duda de que en pocos años afectará a todos, por lo que se ha empezado a utilizar el protocolo IPv6 en muchos países, principalmente en Rusia, Francia, Ucrania, Noruega y Estados Unidos [1].

IPv6 ofrece mejoras en protocolos ya existentes e introduce nuevos protocolos como *Stateless Address Autoconfiguration*, el protocolo de autoconfiguración sin estado que permite a los nodos de un enlace obtener automáticamente direcciones IPv6, aprender sus rutas por defecto y configurar diferentes opciones que se estudiarán a lo largo de este documento.

Dado que los sistemas Windows actualmente vienen con soporte para IPv6 por defecto, pero no implementan completamente el protocolo de autoconfiguración sin estado, se propuso la creación de un servicio que solucione esta problemática.

El objetivo del presente Trabajo Especial de Grado es la creación de un Servidor de Autoconfiguración sin Estado para IPv6 en Plataforma Windows con interfaz gráfica que facilite la configuración de todos los parámetros y opciones soportados en el protocolo Stateless Address Autoconfiguration. Además, se ofrece un completo sistema de depuración a través de un visor de eventos en tiempo real y una herramienta para la administración de las interfaces de red que soporten IPv6.

Para comprender el proceso de desarrollo de la aplicación, este documento se ha dividido en once capítulos que se explican a continuación:

- ✓ **Capítulo 1: El Problema.**

Se plantean las ventajas y desventajas de los sistemas actuales con soporte de IPv6, se justifica la necesidad de implementar la aplicación que se propone, se especifican los objetivos de este trabajo y se establece cual será el alcance de la misma.

- ✓ **Capítulo 2: Marco Teórico.**

Se introducen todos los términos necesarios para la comprensión del trabajo, incluyendo un análisis de todos los RFCs necesarios para la implementación del servicio.

- ✓ **Capítulo 3: Marco Metodológico.**

Se describe la metodología de desarrollo *Scrum* utilizada para la creación de la aplicación. Se definen los conceptos relacionados con el desarrollo del sistema incluyendo los roles de los participantes y etapas del proceso.

- ✓ **Capítulo 4: Marco Aplicativo.**

Se describe la primera fase de la metodología propuesta, se detalla la estructura de la aplicación y se introducen los siguientes capítulos que reflejan los diferentes *Sprints* que se realizaron durante el proceso de desarrollo.

- ✓ **Capítulo 5: Servidor**

Se inicia la segunda fase de la metodología (primer *Sprint*), se analiza la estructura del servidor wradvs y se explican los procesos realizados para permitir la creación del servicio.

- ✓ **Capítulo 6: Herramienta de Configuración del Servidor.**

Se explica el segundo *Sprint*, escritura y lectura del archivo de configuración que utiliza el servidor para su funcionamiento. Se analiza el proceso de registro de eventos para el visor de eventos.

- ✓ **Capítulo 7: Visor de Eventos**

Se analiza el proceso de lectura del log de Windows para la creación de las entradas en la tabla de eventos. Se revisan los fragmentos de código que hacen posible los mensajes detallados de información del tercer *Sprint*.

- ✓ **Capítulo 8: Herramienta de Configuración de Direcciones IPv6.**

Se explica el cuarto módulo de la aplicación en el cuarto *Sprint*, se muestran los procesos para hacer las llamadas a *netsh* y los métodos para realizar la interfaz gráfica a partir de las salidas obtenidas por el programa.

- ✓ **Capítulo 9: Tooltips**

Se muestra la última fase de programación de la aplicación (quinto *Sprint*), como son creados los tooltips, su función y los idiomas que soporta.

- ✓ **Capítulo 10: Pruebas**

Una vez culminada la aplicación se realizan las pruebas necesarias para demostrar que la aplicación funciona de forma correcta.

- ✓ **Capítulo 11: Conclusiones**

Se dan las conclusiones, las recomendaciones y se habla de los futuros proyectos que se pueden realizar en torno a la aplicación propuesta.

1. El Problema

1.1 Planteamiento del Problema

Dado que IPv6 es una tecnología relativamente nueva, muchas de las herramientas que funcionan bajo su entorno poseen deficiencias. Una de estas son los demonios de *Router Advertisement* (RA), que en algunos sistemas Unix pueden estar bien desarrollados, pero que en ningún caso ofrecen interfaz gráfica para los usuarios.

Windows posee una herramienta llamada *netsh* (*Network Shell*), la cual permite configurar un servidor de autoconfiguración (por línea de comandos) para realizar envíos de mensajes RA.

Dicho servidor es muy primitivo y no posee un buen soporte para las diferentes opciones de los mensajes RA. Por esta razón y por el hecho que no tiene la interfaz gráfica característica de los sistemas Windows, se ha decidido implementar un servicio que soporte completamente el envío de mensajes RA y sus opciones. Este servicio deberá ser configurado mediante una interfaz gráfica amigable. Adicionalmente será incluida una herramienta de configuración de las interfaces de red que soportan IPv6.

1.2 Objetivos

A continuación se describen el objetivo general y los objetivos específicos del trabajo especial de grado.

1.2.1 Objetivo General

Desarrollar un *Servicio de Windows* que permita la autoconfiguración sin estado de los nodos de un enlace IPv6. La configuración de los parámetros del servicio se realizará mediante una interfaz gráfica amigable.

1.2.2 Objetivos Específicos

- ✓ Desarrollar el servidor de RA con soporte para todos los parámetros de configuración involucrados en el envío de mensajes RA.
- ✓ Diseñar una interfaz gráfica amigable para plataforma Windows con el fin de realizar la configuración del servicio.
- ✓ Validar que los datos suministrados por el usuario cumplan con las especificaciones dadas por los RFCs y mostrar mensajes de error descriptivos en caso de incongruencias.

- ✓ Realizar un módulo de configuración básica para las interfaces de red con soporte de IPv6.
- ✓ Permitir al usuario visualizar un registro de los eventos que ocurren en el proceso de autoconfiguración en tiempo real, dando la opción de guardar los eventos en un documento externo y en diferentes formatos.

1.3 Justificación

Una de las características de IPv6 es tratar de simplificar el proceso de configuración de los hosts. La autoconfiguración simplifica el trabajo de los administradores de red ya que los hosts obtienen sus parámetros (direcciones, rutas, routers por defecto, servidores DNS) de red automáticamente. Sin embargo, en el router o bien en los dispositivos que utilizan algún demonio de RA hay que realizar una mínima configuración. Esta configuración puede llegar a ser compleja para usuarios sin experiencia dado que no se ofrecen suficientes herramientas que faciliten el proceso.

1.4 Alcance

El servidor de autoconfiguración se ejecutará en sistemas Windows (XP y Vista). Permitirá la configuración de direcciones IPv6 en las diferentes interfaces de red existentes. Los nodos vecinos al servidor de configuración se autoconfigurarán gracias a los mensajes *Router Advertisement*.

2. Marco Teórico

2.1 Internet Protocol version 6 (IPv6)

IPv6 es el protocolo diseñado por el IETF (*Internet Engineering Task Force*) para reemplazar a su predecesor IPv4.

IPv4 es un protocolo que tiene más de veinte años y aun cuando es bastante funcional, actualmente empieza a tener algunas deficiencias, como la escasez de direcciones en algunos países[1].

2.1.1 Características de IPv6

IPv6 es un protocolo con nuevas características y algunas mejoras sobre IPv4. Las características más relevantes se mencionan a continuación:

- ✓ **Direccionamiento extendido**

Una de las principales características de IPv6 es que ataca el problema de escasez de direcciones IPv4, para lo cual se incrementa el tamaño de las direcciones de 32 bits (IPv4) a 128 bits (IPv6).

- ✓ **Simplificación de la cabecera**

Aun cuando el encabezado aumenta de tamaño, se ha simplificado y pasa de tener 13 campos (IPv4) a tener 8 campos (IPv6).

- ✓ **Soporte para opciones mejorado**

En IPv4 el espacio para opciones va de 0 a 40 bytes, mientras que en IPv6 se maneja con encabezados de extensión lo cual permite encadenar las opciones y no tener un límite de espacio.

- ✓ **Direccionamiento jerárquico**

Se mejoran las capacidades de enrutamiento dando mayores facilidades en el proceso de agregación.

- ✓ **Direcciones multicast mejoradas**

Ya no existen las direcciones de *broadcast*. Si es necesario enviar un mensaje a todos los nodos de una misma subred, se hace a través de direcciones multicast. Este mecanismo ya existía en IPv4 pero ahora toma fuerza y las direcciones multicast de IPv6 tienen un alcance.

[1] http://www.ipv6.org

✓ **Direcciones anycast**
Además de hacer uso intensivo de las direcciones *multicast*, IPv6 introduce el concepto de direcciones anycast. Estas nuevas direcciones sirven para enviar un paquete al nodo más cercano de un grupo determinado.

✓ **Mejor soporte de QoS**
En IPv4 existe un campo de 8 bits (Type of Service) para soportar QoS. En IPv6 se tienen dos campos para este propósito (Traffic Class y Flow Label).

✓ **Fragmentación en el emisor**
La fragmentación de paquetes se utiliza para enviar un paquete cuyo tamaño es mayor al MTU (Maximum Transmission Unit). En IPv4 la fragmentación se puede dar tanto en el emisor como en cualquier dispositivo intermedio de capa de red. En IPv6, sólo ocurre en el emisor lo cual disminuye la carga de los routers.

✓ **Soporte para movilidad**
Se refiere a la posibilidad que tiene un equipo de mantener la misma dirección IP aun después de cambiar de red. En IPv4 existe un proceso que realiza esta función, sin embargo en IPv6 se crea un protocolo especial para manejar esta situación.

✓ **Autoconfiguración**
Otro aspecto muy importante de IPv6 es que está hecho para permitir la autoconfiguración, es decir, que cada dispositivo final de una red podrá obtener automáticamente sus direcciones IPv6.

2.1.2 Cabecera IPv6
La cabecera IPv6 consta de 40 bytes divididos en ocho campos. Los campos de la cabecera IPv6 se muestra en la Figura 2.1 (tomada de [2]):

✓ **Version**
El campo de 4 bits *Version* indica la versión del protocolo de Internet que se utiliza, en este caso el valor del campo siempre debe ser 6.

✓ **Traffic Class**
El campo de 8 bits *Traffic Class* facilita el manejo de datos en tiempo real o que requieran cualquier tipo de trato especial. Los nodos emisores y routers intermedios pueden utilizarlo para identificar y distinguir entre diferentes clases o prioridades de paquetes IPv6. Este campo es equivalente al campo *Type of Service* de IPv4 [3].

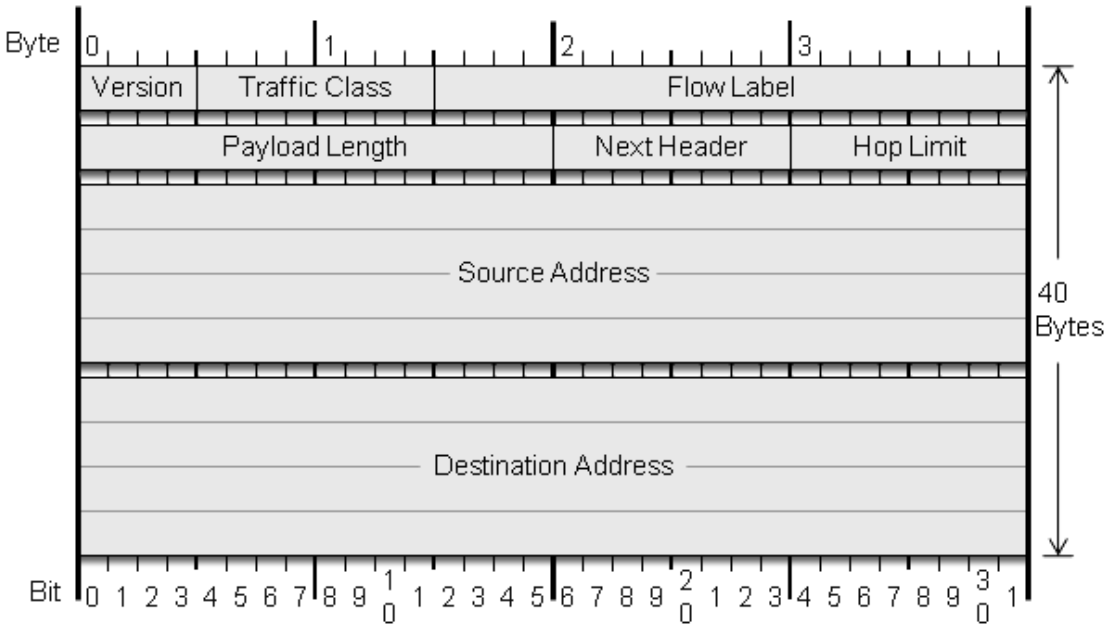

Figura 2.1: Cabecera IPv6.

✓ **Flow Label**

El campo de 20 bits *Flow Label* distingue paquetes que poseen el mismo procesamiento con el fin de mejorar el tráfico en tiempo real. Un emisor etiqueta secuencias de paquetes con diferentes opciones. Los routers pueden procesar paquetes de la misma secuencia de forma más eficiente dado que no deben reprocesar su cabecera [3].

✓ **Payload Length**

El campo de 16 bits *Payload Length* indica cual es la longitud en bytes de la carga útil, incluyendo las cabeceras de extensión. El valor debe ser un entero sin signo [2].

✓ **Next Header**

El campo de 8 bits *Next Header* contiene el número que identifica el protocolo o cabecera de extensión que sigue inmediatamente después de la cabecera IPv6.

✓ **Hop Limit**

El campo de 8 bits *Hop Limit* especifica el número máximo de routers por los cuales puede ser transmitido un paquete antes de ser descartado. Es un contador que va en decremento a medida que el paquete es reenviado [4].

✓ **Source Address**
El campo de 128 bits *Source Address* indica la dirección IPv6 del emisor del mensaje.

✓ **Destination Address**
El campo de 128 bits *Destination Address* representa la dirección IPv6 del receptor del mensaje. Si la cabecera de extensión *Routing Header* está presente, es posible que la dirección de destino no sea el nodo final sino algún nodo intermedio.

2.1.3 Cabeceras de extensión

En IPv4 existe un campo de opciones que comienza en el byte 20, que puede medir entre 0 y 40 bytes. Este campo, cuando existe, es verificado en todos los nodos desde el origen hasta el destino, lo que produce mayor latencia (ya que habrá más procesamiento en cada nodo). Para mejorar el manejo de opciones IPv6 utiliza cabeceras de extensión, las cuales no son verificadas en los nodos intermedios (a excepción de la opción *Hop-by-Hop*), reduciendo la latencia.

Existen seis cabeceras de extensión distintas:

- Hop-by-Hop Options
- Routing
- Fragment
- Destination Options
- Authentication
- Encapsulating Security Payload

Un paquete IPv6 puede contener cero o más cabeceras de extensión localizadas entre la cabecera IPv6 y la cabecera del protocolo de capa superior. Cada cabecera de extensión es identificada por el campo *Next Header* de la cabecera previa. Véase la Figura 2.2.

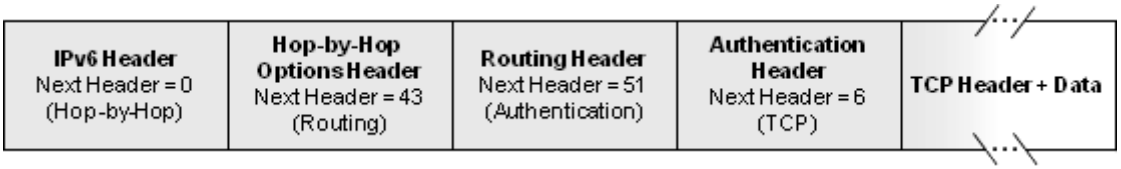

Figura 2.2: Cabeceras de extensión.

La longitud de las cabeceras de extensión debe ser un número entero múltiplo de 8 bytes. En caso de que la opción no cumpla con dicho requerimiento se utilizarán bits de relleno para lograrlo [4].

Las cabeceras de extensión son procesadas en el orden en que se presentan. Los nodos intermedios sólo toman en cuenta la opción *Hop-by-Hop* por lo tanto ésta debe ser la primera cabecera de extensión. Adicionalmente el RFC 2460 recomienda utilizar el siguiente orden para concatenar las cabeceras:

1. IPv6 header
2. Hop-by-Hop Options header
3. Destination Options header (para opciones que serán procesadas en el primer destino que aparece en el campo *Destination Address*, más los destinos subsecuentes listados en el *Routing Header* [2])
4. Routing header
5. Fragment header
6. Authentication header
7. Encapsulating Security Payload header
8. Destination Options header (para opciones que serán procesadas sólo por el destinatario final del paquete)
9. Upper Layer header (Cabecera de capa superior)

A continuación se explicarán las seis cabeceras de extensión:

✓ **Hop-by-Hop Options Header**

Esta cabecera posee información opcional que debe ser examinada por cada uno de los nodos a lo largo del camino del paquete. La cabecera *Hop-by-Hop* debe venir inmediatamente después de la cabecera IPv6, la cual utiliza el valor del campo *Next Header* en cero para indicarlo [3].

La cabecera *Hop-by-Hop* posee tres campos que son: *Next Header*, *Header Extension Length* y *Options* (que puede ser de tamaño variable).

El campo de 8 bits *Next Header* indica cual será la cabecera que sigue después de procesadas las opciones.

El campo de 8 bits *Header Extension Length* indica cual será el tamaño del siguiente campo en unidades de 8 bytes. Si la siguiente opción utiliza menos de 64 bits, entonces el valor del campo *Header Extension Length* será cero.

Puede haber una o más opciones en el campo *Options*. El tamaño en el campo *Options* es variable y es determinado según el campo *Header Extension Length.*

Dos de las opciones más destacadas son:

- **Type Jumbogram**
  El campo *Payload Length* de la cabecera IPv6 ocupa 16 bits, esto significa que la carga útil máxima de un paquete es 2^{16} bytes. Cuando se desea usar la opción *Jumbo Payload* (*Hop-by-Hop Options Type* = 194) el valor del campo *Payload Length* se coloca en cero y se otorga un nuevo campo para colocar el tamaño de la carga útil. Este nuevo campo es de 32 bits, permitiendo así una carga útil máxima de 2^{32}-1 bytes.

- **Router Alert**
  Le indica a los nodos intermedios que el paquete contiene información importante para el momento de reenviar. Se utiliza mayormente por los protocolos *Multicast Listener Discovery* y *Resource Reservation Protocol* [3].

✓ **Routing Header**

Esta cabecera es usada por un emisor IPv6 para listar uno o más nodos que deben ser "visitados" en el camino hacia el destino. El *Routing Header* se identifica en el campo *Next Header* con el valor 43 [3].

El *Routing Header* posee cinco campos: *Next Header*, *Header Extension Length*, *Routing Type*, *Segments Left* y *Type-Specific Data*. *Next Header* y *Header Extension Length* cumplen la misma función que en el Hop-*by-Hop Options Header*. El campo T*ype-Specific Data* es de tamaño variable y múltiplo de 8 bytes. Este campo depende del valor que indique el campo *Routing Type*. En caso de que el valor del *Routing Type* sea desconocido por el nodo que lo recibe, deberá actuar dependiendo del valor que se indica en el campo *Segments Left*. Si este campo no contiene los nodos que deben ser visitados se debe ignorar el *Rounting Header* y procesar la siguiente cabecera, si el campo *Segments Left* no es cero, el nodo debe descartar el paquete y enviar un mensaje ICMPV6 de tipo *Parameter Problem* con el campo *Code* en cero [3].

✓ **Fragment Header**

Si un emisor IPv6 desea enviar un paquete a su destinatario, debe utilizar el protocolo *Path MTU discovery* para determinar el tamaño máximo que puede tener el paquete a lo largo del camino hacia el destino. Si el paquete es más

grande que el MTU (Maximum Transfer Unit) soportado, entonces se fragmenta. En IPv6 a diferencia de IPv4 la fragmentación se lleva a cabo sólo en el origen y no en los nodos intermedios. El receptor sabe cómo manejar los fragmentos para reensamblarlos. El valor del campo *Next Header* correspondiente a esta opción es 44.

✓ **Destination Options Header**

La cabecera *Destination Options* es usada para llevar información opcional que debe ser examinada en los destinatarios. Esta cabecera se identifica con el valor 60 en el campo *Next Header* de la cabecera previa.

La cabecera *Destination Options* puede aparecer dos veces en un paquete IPv6, cuando se inserta antes del *Routing Header* contiene información que debe ser procesada en la lista de routers que ofrece. Cuando la cabecera *Destination Options* se inserta antes del protocolo de capa superior, la información debe ser procesada en el nodo destinatario.

✓ **Encapsulating Security Payload (ESP)**

El propósito de esta opción es encriptar la información que se envía en el paquete IP. El encriptado se obtiene aplicando transformaciones especificas en los datos que se quieren proteger. Los datos se pueden empezar a encriptar en cualquier parte después de la cabecera IP y antes de la cabecera de capa de transporte [5].

✓ **Authentication Header (AH)**

Esta cabecera es utilizada para proveer integridad y autenticación a los datos de origen para el datagrama IP. La información de autenticación es calculada usando todos los campos del datagrama que no varían durante la transmisión. Esta cabecera debe ser colocada después del *Fragment Header* y antes del protocolo de capa de transporte. Si se implementa ESP, el *Authentication Header* debe ir antes del ESP.

2.1.4 Direccionamiento en IPv6

A continuación se profundizará en los aspectos relacionados con las direcciones IPv6, tales como su representación y sus diferentes tipos. El RFC 4291 define estos aspectos.

✓ **Notación de las direcciones IPv6**

El tamaño de una dirección IPv6 es de 128 bits, lo que quiere decir que se tiene un máximo de 2^{128} direcciones disponibles. Para su representación, los 128 bits se dividen en bloques de 16 bits y cada uno de estos es convertido a cuatro dígitos hexadecimales separados por el símbolo dos puntos (“:”) [4].

Una dirección IPv6 puede ser simplificada eliminando los ceros no significativos de un bloque, siempre dejando al menos un dígito.

Finalmente es posible realizar una compresión de ceros, reemplazando bloques de ceros consecutivos con doble dos puntos ("::"). En la Figura 2.3 se puede apreciar el proceso completo.

Figura 2.3: Representación de dirección IPv6.

✓ **Notación de los prefijos en IPv6**

Un prefijo son los bits más significativos de una dirección IPv6, los cuales son usados para identificar una subred o un tipo de dirección en específico. La notación es la siguiente: <dirección IPv6>/<longitud del prefijo>.

En la práctica, todas las subredes poseen un prefijo de 64 bits, es por ello que cuando se habla de una dirección *unicast* no es necesario indicar la longitud.

✓ **Direcciones Unicast**

Es un identificador para una interfaz única. Los paquetes dirigidos a una dirección de este tipo son entregados solamente a la interfaz identificada con esta dirección.

- **Global Unicast**

  Son direcciones globales identificadas con un formato de prefijo iniciado por los bits 001. Las direcciones globales de IPv6 son equivalentes a las direcciones públicas de IPv4, son enrutables mundialmente y alcanzables en Internet6.

  Su estructura está definida en el RFC 4291 y se puede apreciar en la Figura 2.4 (tomada de [6]).

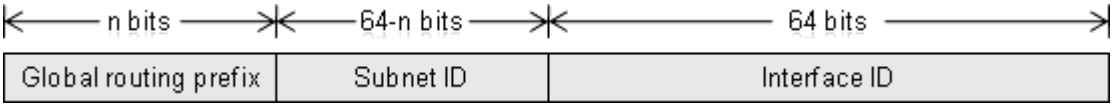

Figura 2.4: Estructura de la dirección *Global Unicast*.

El campo *Global routing prefix* identifica el rango de direcciones asignado por los RIR[2] y el proveedor de Internet (ISP). El campo *Subnet ID* identifica el enlace o subred. Por último, el campo *Interface ID* identifica una interfaz en la subred y debe ser única para esa subred.

El prefijo 2001:DB8::/64 se encuentra reservado para ser utilizado con fines documentales.

- **Direcciones especiales**

La primera parte del espacio de direcciones de IPv6 cuyo prefijo es 0x00 está reservada. De la cual se han definido dos direcciones especiales mencionadas a continuación:

- **Unspecified (no especificada)**

La dirección no especificada es usada solamente para indicar la ausencia de dirección válida. Es comúnmente usada como dirección fuente cuando una dirección no ha sido asignada aún. Esta dirección posee el valor 0:0:0:0:0:0:0:0, por lo cual aplicando el método de compresión de ceros se puede abreviar como ::.

- **Loopback**

La dirección loopback 0:0:0:0:0:0:0:1 es usada para identificar una interfaz loopback, permitiendo enviar paquetes a sí mismo sin salir a la red, lo cual es conveniente para comprobar el correcto funcionamiento de la pila IPv6. Se puede abreviar como ::1.

- **Direcciones IPv4 embebidas en Direcciones IPv6**

Fueron definidas dos tipos de direcciones para facilitar la migración de IPv4 a IPv6 y la coexistencia de ambos tipos de hosts. Son descritas en el RFC 4291.

IPv4-compatible

Son usadas para establecer una comunicación con IPv6 a través de una infraestructura de IPv4 que utiliza direcciones públicas, tal como Internet. Este tipo de direcciones es muy raramente usado y se

[2] Regional Internet Registry: es una organización que supervisa la asignación y el registro de recursos de direcciones IPv4 e IPv6 dentro de una región particular del mundo.

considera obsoleto. La Figura 2.5 (tomada de [6]) muestra la estructura de este tipo de direcciones.

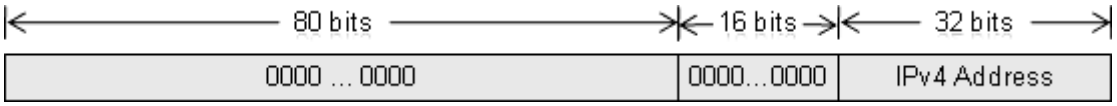

Figura 2.5: Estructura de la dirección *IPv4-compatible IPv6*.

IPv4-mapped

Es un tipo de dirección usado para representar las direcciones de nodos IPv4 como direcciones IPv6. Un nodo IPv6 puede usar esta dirección para enviar un paquete a un nodo IPv4. La Figura 2.6 (tomada de [6]) muestra la estructura de este tipo de direcciones.

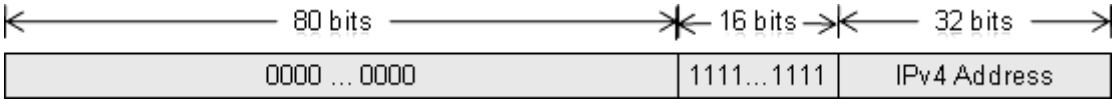

Figura 2.6: Estructura de la dirección *IPv4-mapped IPv6*.

- **Direcciones 6to4**

El uso de las direcciones 6to4 es uno de los mecanismos diseñados para permitir a hosts o redes IPv6 comunicarse a través de una infraestructura que sólo soporta IPv4. En la Figura 2.7 (tomada de [7]) se puede observar la estructura de este tipo de direcciones.

Figura 2.7: Estructura de la dirección 6to4.

El prefijo comienza con 0x2002, seguido de la dirección pública IPv4, formando así un prefijo que se puede abreviar como 2002:DirIPv4::/48 [7].

Posteriormente se encuentra el campo *SLA ID* de 16 bits, el cual identifica la subred. Por último el campo *Interface ID* de 64 bits que indica el identificador de interfaz.

- **Link-Local**

Se identifican por el prefijo FE80::/64. Son usadas por los nodos al comunicarse con sus vecinos en el mismo enlace, su alcance es la subred y nunca deben ser enrutadas. Pueden ser utilizadas en mecanismos de

autoconfiguración, en *Neighbor Discovery* (sección 2.3) y en redes sin routers, por lo tanto son bastante útiles en la creación de redes temporales. En la Figura 2.8 (tomada de [6]) se muestra la estructura:

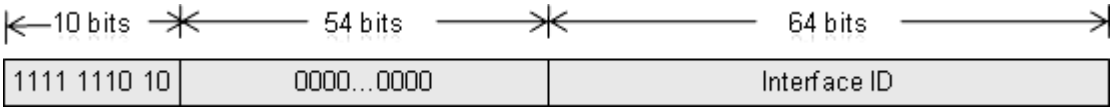

Figura 2.8: Estructura de la dirección Link-local.

Generalmente son autoconfiguradas y derivadas de la dirección de capa de enlace, aun cuando existe la posibilidad de configurarlas en forma manual.

El proceso para obtener una dirección link-local es (véase la Figura 2.9):

1. Obtener el EUI-64 a partir de la dirección MAC, insertando los bytes 0xFF y 0xFE entre el tercer y cuarto byte.
2. Se invierte el valor del bit *u*, el cual está situado en la séptima posición más significativa. Obteniendo así la dirección EUI-64 modificada (identificador de interfaz) [6].
3. Se antepone el prefijo link-local FE80::/64 a la dirección obtenida en el paso previo.

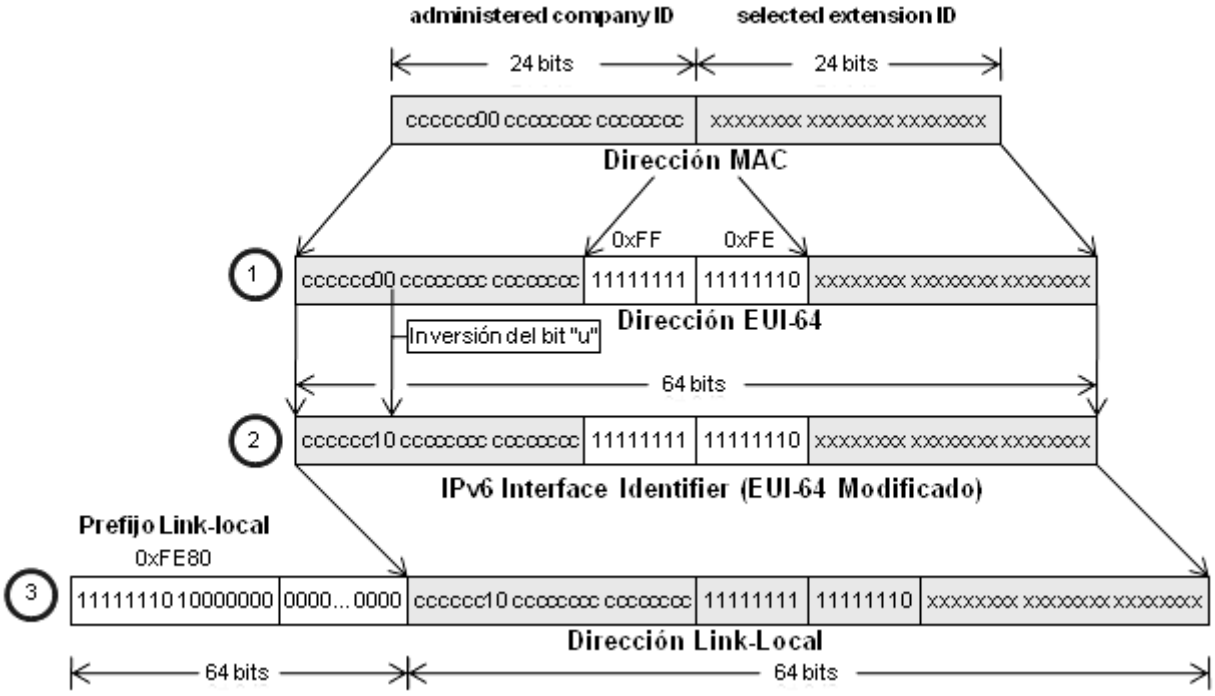

Figura 2.9: Obtención de dirección Link-local.

- **Site-local**

Las direcciones site-local son las equivalentes al espacio de direcciones privadas de IPv4. Estas direcciones no son alcanzables desde otras redes y los routers no deben retransmitir este tipo de tráfico fuera de la red. El prefijo de una dirección site-local es FEC0::/10, su estructura se muestra en la Figura 2.10 (tomada de [6]):

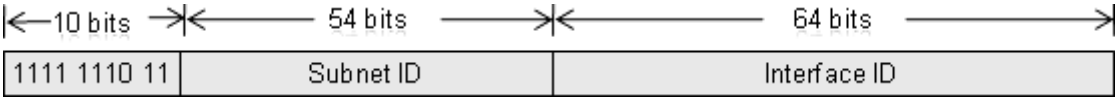

Figura 2.10: Estructura de la dirección Site-local.

Este tipo de direcciones se consideran obsoletas (RFC 3879). Fueron reemplazadas por las direcciones Unique-local.

- **Unique-local**

Estas direcciones son únicas globalmente pero no deben ser enrutadas hacia Internet. Comúnmente llamadas *local IPv6 address* (dirección IPv6 local), son especificadas en el RFC 4193. Su estructura se puede apreciar en la Figura 2.11 (tomada de [8]):

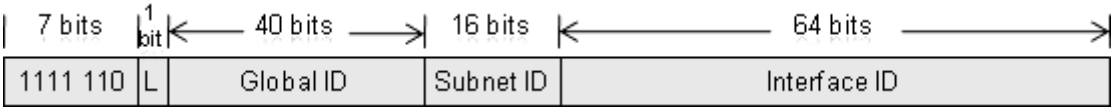

Figura 2.11: Estructura de la dirección Unique-local.

Utilizan el prefijo FC00::/7. El bit *L* con valor 1 indica que el prefijo es asignado localmente, el uso del bit en 0 será definido en el futuro. Los siguientes 40 bits correspondientes al campo *Global ID* son usados para crear un prefijo único globalmente, el cual es generado de forma aleatoria. Finalmente, los últimos 80 bits correspondientes a los campos *Subnet ID* e *Interface ID*, identifican la subred y la interfaz respectivamente.

En consecuencia de que el bit *L* por ahora siempre posee valor 1, el prefijo toma el valor FD00::/8.

✓ **Direcciones Multicast**

Una dirección *multicast* es un identificador para un grupo de nodos. Un nodo IPv6 puede estar a la escucha de múltiples direcciones *multicast*. Los nodos pueden unirse o dejar un grupo *multicast* en cualquier momento. Las direcciones *multicast* no pueden ser utilizadas como direcciones origen ni como un destino intermedio mediante el uso de la cabecera de extensión *Routing header*.

Las direcciones *multicast* poseen el prefijo FF, seguido de 3 campos. La Figura 2.12 (tomada de [6]) muestra la estructura de una dirección *multicast*.

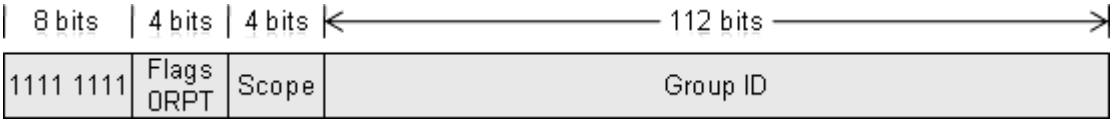

Figura 2.12: Estructura de la dirección *Multicast*.

El valor de los primeros 8 bits es 1. Los siguientes 4 bits son banderas utilizadas de la siguiente forma: el primer bit está reservado para uso futuro y debe ser cero 0. El segundo bit (*R*) indica si la dirección *multicast* embebe un *Rendezvous Point*[3]. El tercer bit (*P*) en 1 indica que la dirección *multicast* posee información de prefijo. Por último el bit *T* en 1 indica si la dirección es temporal y en cero indica si es una dirección asignada por la IANA.

El campo *Scope* es usado para determinar el alcance de la dirección *multicast*. Los posibles valores se encuentran en la Tabla 2.1.

Valor	Alcance (Scope)	Descripción
1	Interface-local	Extiende su alcance a la interfaz de un nodo.
2	Link-local	Su alcance es dentro del enlace.
4	Admin.-local	El alcance más corto que debe ser administrativamente configurado.
5	Site-local	Su alcance permanece dentro del sitio.
8	Organization-local	Su alcance son múltiples sitios pertenecientes a una misma organización.
6, 7, 9 A, B, C, D	Unassigned	No han sido asignados.
E	Global	Poseen alcance global.
0, 3, F	Reserved	Están reservados, no se permite su uso.

Tabla 2.1: Valores del campo *Scope*.

[3] Rendezvous Point: un punto de distribución para un flujo multicast en específico.

- **Direcciones multicast conocidas**

El RFC 2375 define direcciones *multicast* que han sido asignadas permanentemente, la Tabla 2.2 muestra algunas de estas direcciones.

Dirección	Descripción
FF01::1	Todos los nodos en la interfaz.
FF01::2	Todos los routers en la interfaz.
FF02::1	Todos los nodos en el enlace.
FF02::2	Todos los routers en el enlace.
FF05::2	Todos los routers en el sitio.
FF02::1:2	Todos los agentes DHCPv6 en el enlace.
FF05::1:3	Todos los servidores DHCPv6 en el sitio

Tabla 2.2: Algunas direcciones *multicast* conocidas.

- **Solicited-Node Multicast Address**

Es una dirección *multicast* que cada nodo debe tener por cada dirección *unicast* y *anycast* que posee. Es usada en el protocolo *Neighbor Discovery*. El RFC 4291 define este tipo de direcciones.

Las direcciones *solicited-node* se obtienen tomando los 24 bits menos significativos de la dirección IPv6 y concatenándolos al prefijo FF02::1:FF00:0/104. Un ejemplo se puede apreciar en la Figura 2.13.

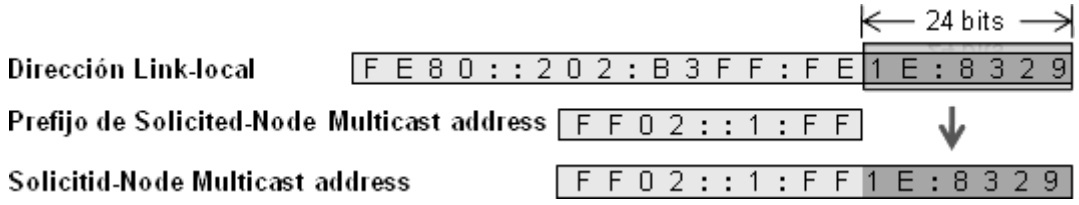

Figura 2.13: Ejemplo de dirección *multicast Solicited-Node*.

- **Dirección link-layer asociada a un grupo multicast**

Una dirección link-layer se refiere a la dirección de capa de enlace, por ejemplo una dirección MAC. El prefijo utilizado para las direcciones link-layer *multicast* es 33:33. El resto de la dirección son los últimos 4 bytes de la dirección IPv6 *multicast*. Por ejemplo, la IPv6 *multicast* FF02::1:FF1E:8329 tiene asociada la dirección link-layer *multicast* 33:33:FF:1E:83:29.

✓ **Direcciones Anycast**

Son un intermedio entre las direcciones *unicast* y *multicast*. Usan el espacio de direcciones *unicast* y los paquetes siempre son enviados a una sola interfaz. Pero hay varias interfaces escuchando con esa dirección, como es el caso de *multicast* [9].

Las direcciones *anycast* se encuentran aún en desarrollo. Hasta ahora sólo es soportada la dirección *anycast Subnet Router*. Esta direcciona a todos los routers de una subred. La dirección se representa con el prefijo de subred seguido de ceros. Por ejemplo, la dirección *anycast* 2001:db8:1234:5678:: corresponde a los routers de la subred 2001:db8:1234:5678::/64.

2.2 Internet Control Message Protocol version 6 (ICMPv6)

ICMPv6 es usado por los nodos IPv6 para el manejo de información y reportar errores encontrados en los paquetes que se procesan. La función principal de este protocolo es diagnosticar el estado de una red. ICMPv6 debe ser implementado por completo en todos los nodos IPv6 [10].

2.2.1 Formato general del mensaje

Cada mensaje ICMPv6 es precedido por un encabezado IPv6 y cero o más cabeceras de extensión. La cabecera ICMPv6 es identificada en el campo *Next Header* de la cabecera anterior con el valor 58. Este valor es diferente al utilizado para identificar ICMPv4.

El formato del mensaje ICMPv6 se observa en la Figura 2.14 (tomada de [10]) a continuación:

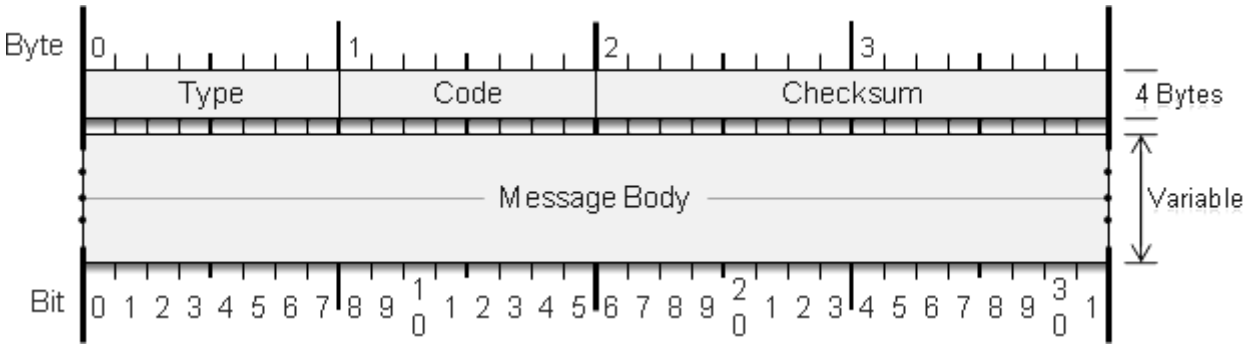

Figura 2.14: Formato general del mensaje ICMPv6.

✓ Type

El campo de 8 bits *Type* indica el tipo de mensaje ICMPv6 que se enviará. Este campo va a determinar el formato del mensaje ICMPv6.

✓ Code

El campo de 8 bits *Code* depende del tipo de mensaje indicado por el campo *Type*. Es utilizado para otorgar un nivel de clasificación adicional a los mensajes.

✓ Checksum
El campo de 16 bits *Checksum* es para detectar errores de los datos de los mensajes ICMPv6 y parte de la cabecera IPv6.

✓ Message Body
El campo *Message Body* depende del valor del campo *Type*.

2.2.2 Tipos de Mensajes
Los mensajes ICMPv6 se dividen en dos grupos, mensajes de error y mensajes de información. Los mensajes de error son identificados en el campo *Type* con un cero en el bit más significativo, es decir que utiliza valores entre 0 y 127. Los mensajes de información toman valores entre 128 y 255 en el campo *Type* [10].

✓ **Mensajes de error**
En la Tabla 2.3 se listan los principales tipos de mensajes de error especificados en el RFC 4443 (ICMPv6).

Type	Descripción
1	Destination Unreachable
2	Packet Too Big
3	Time Exceded
4	Parameter Problem
100	Para experimentos privados
101	Para experimentos privados
127	Reservado para expandir los mensajes de error ICMPv6

Tabla 2.3: Mensajes de Error ICMPv6.

- **Destination Unreachable**

El mensaje de destino inalcanzable (*Destination Unreachable*) debe ser generado por un router o por el nodo origen, en respuesta a paquetes que no pueden ser entregados al destino por razones que no sean congestión en la red. El formato del mensaje está dado en la Figura 2.15 (tomada de [10]):

Este mensaje de error es identificado con el valor 1 en el campo *Type*. El campo *Code* (ver Tabla 2.4) indica el motivo por el cual el mensaje no puede llegar al destino, es decir, que tipo de mensaje de error se debe enviar. El campo *Length* definido en el RFC 4884 [11] indica el tamaño del campo *Data* que contiene la información del datagrama original (el que generó el error en primer lugar). Este campo surge como solución a un problema que ocurre cuando el mensaje ICMPv6 posee alguna estructura de extensión adicional adjunta, en cuyo caso de no conocer el tamaño del

campo *Data* no se puede saber en qué punto del mensaje comienza la siguiente estructura por lo que habrá errores en el procesamiento del paquete.

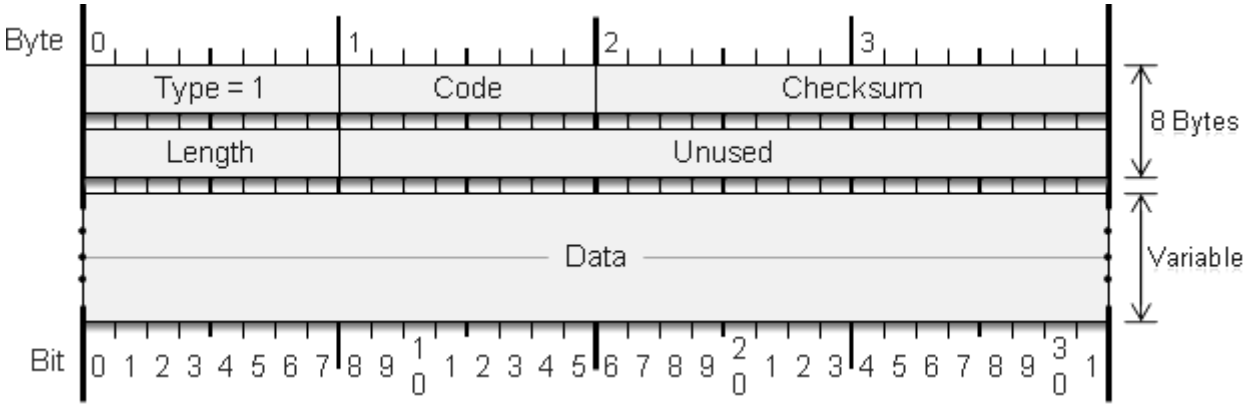

Figura 2.15: Formato del mensaje *Destination Unreachable*.

El campo *Unsused* toma el valor cero y no se ha definido su uso aún. El campo *Data* posee la cabecera IPv6 del paquete que originó el mensaje ICMPv6, incluyendo los datos pero sin exceder el MTU.

Code	Descripción
0	No hay ruta al destino
1	Comunicación prohibida con el destino
2	Fuera del alcance de la dirección destino.
3	Dirección inalcanzable
4	Puerto inalcanzable
5	Fallas por políticas de filtrado
6	Rechazada la ruta al destino

Tabla 2.4: Códigos para un mensaje *Destination Unreachable*

- **Packet Too Big**

Este mensaje de error es enviado por un router en respuesta a un paquete que no puede ser transmitido porque el tamaño del mismo es más grande que el MTU del enlace de salida.

El valor del campo *Type* es 2 para este tipo de mensaje. El campo *Code* (ver Figura 2.16, tomada de [10]) es colocado en cero por el emisor y es ignorado por el receptor [10]. El campo *MTU* indica la unidad máxima de transferencia del enlace de salida.

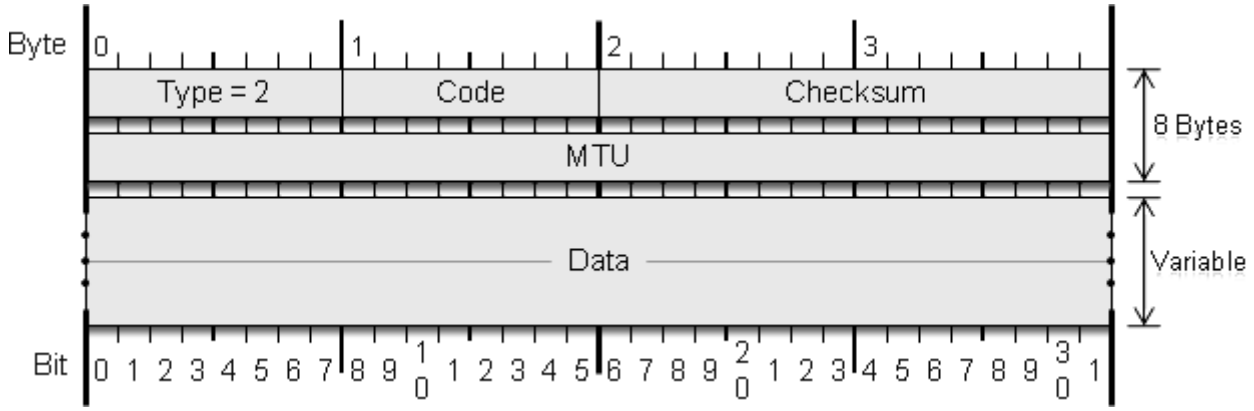

Figura 2.16: Formato del mensaje *Packet Too Big*.

- **Time Exceded**

El campo *Type* con el valor 3 indica que el mensaje de error es *Time Exceded*. El formato del mensaje se puede ver en la Figura 2.17 (tomada de [10]):

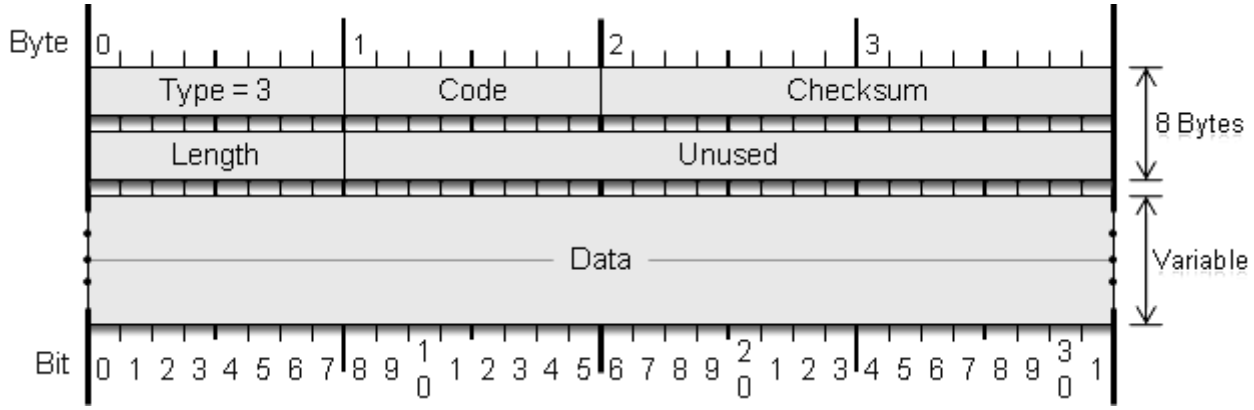

Figura 2.17: Formato del mensaje *Time Exceded*.

Si un router recibe un paquete con el valor del campo *Hop Limit* en cero, o si el router decrementa el valor del campo *Hop Limit* a cero debe entonces enviar un mensaje ICMPv6 de tiempo excedido con el valor del campo *Code* en cero. Cuando se excede el tiempo de reensamblaje de un paquete, el campo *Code* debe tomar el valor 1. El campo *Length* indica el tamaño del campo *Data* que contiene la información del datagrama original

- **Parameter Problem**

Cuando un nodo recibe un paquete con problemas en algún campo de la cabecera IPv6 o alguna cabecera de extensión que no permiten que se complete el procesamiento del paquete, éste se debe descartar y se debe

enviar un mensaje ICMPv6 de tipo 4 (*Parameter Problem*). El formato del mensaje se observa en la Figura 2.18 (tomada de [10]) a continuación:

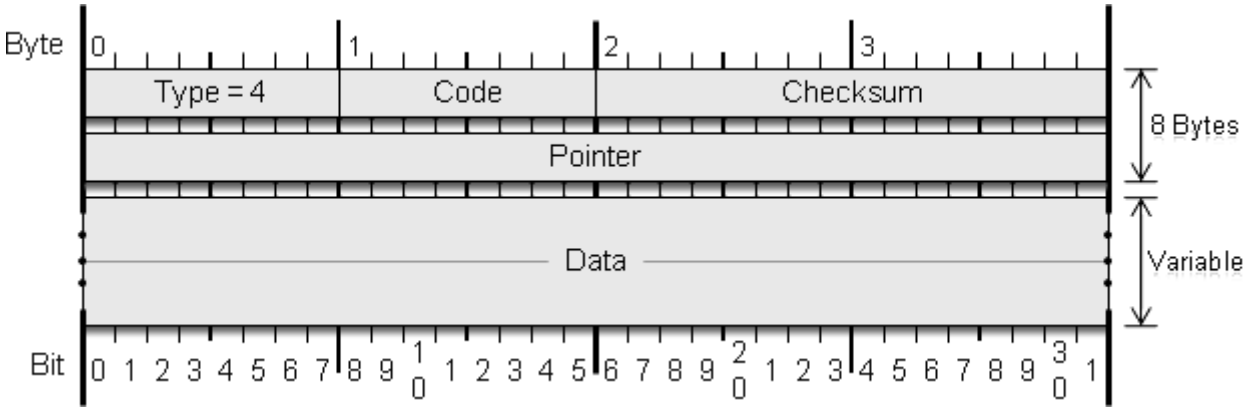

Figura 2.18: Formato del mensaje *Parameter Problem*.

El campo *Code* indica en que parámetro existe un problema (ver Tabla 2.5). El campo *Pointer* indica el byte en el que se encuentra el error. Por ejemplo, un mensaje ICMPv6 de tipo 4 (*Type* = 4), código 1 (*Code* = 1) y *Pointer* en 40, indica que el campo *Next Header* de la primera cabecera de extensión posee un valor desconocido.

Code	Descripción
0	Error en un campo de la cabecera
1	El campo *Next Header* posee un tipo de encabezado desconocido
2	Opción IPv6 desconocida

Tabla 2.5: Códigos para un mensaje *Parameter Problem*

✓ **Mensajes de información**

En la Tabla 2.6 (tomada de [3]) se listan los diferentes mensajes de información.

Los tipos 133,134, 135, 136 y 137 son utilizados en el protocolo *Neighbor Discovery* descrito en la sección 2.3.

Los mensajes *Echo Request* y *Echo Reply* son usados en una de las herramientas más comunes de TCP/IP: *Packet Internet Groper* (PING). PING es usado para determinar si un host en específico se encuentra en la red y listo para comunicarse. El origen emite un mensaje *Echo Request* al destino y si el destino se encuentra disponible contesta con un mensaje *Echo Reply* [3].

Type	Descripción
128	Echo Request
129	Echo Reply
133	Router Solicitation
134	Router Advertisement
135	Neighbor Solicitation
136	Neighbor Advertisement
137	Redirect Message
200	Para experimentos privados
201	Para experimentos privados
255	Reservado para expandir los mensajes de información ICMPv6

Tabla 2.6: Mensajes de Información ICMPv6.

- **Echo Request**

El formato del mensaje *Echo Request* se puede ver en la Figura 2.19 (tomada de [10]). El campo de 8 bits *Type* debe tener el valor 128. El campo de 8 bits *Code* se mantiene con el valor cero. Los campos *Identifier* y *Sequence number* se utilizan para asociar un *Echo Request* a su respectivo *Echo Reply* y el campo *Data* posee cero o más bytes de datos arbitrarios.

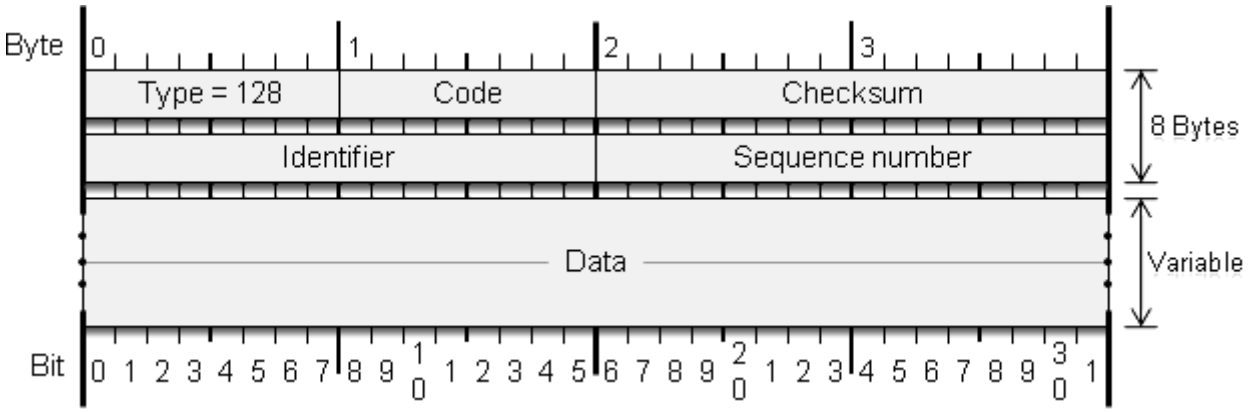

Figura 2.19: Formato del mensaje *Echo Request*.

- **Echo Reply**

El formato del mensaje *Echo Reply* se puede ver en la Figura 2.20 (tomada de [10]). El campo de 8 bits *Type* debe tener el valor 129. El campo de 8 bits *Code* se mantiene con el valor cero. Los campos *Identifier*, *Sequence number* y *Data* deben poseer los mismos valores que el mensaje *Echo Request* recibido.

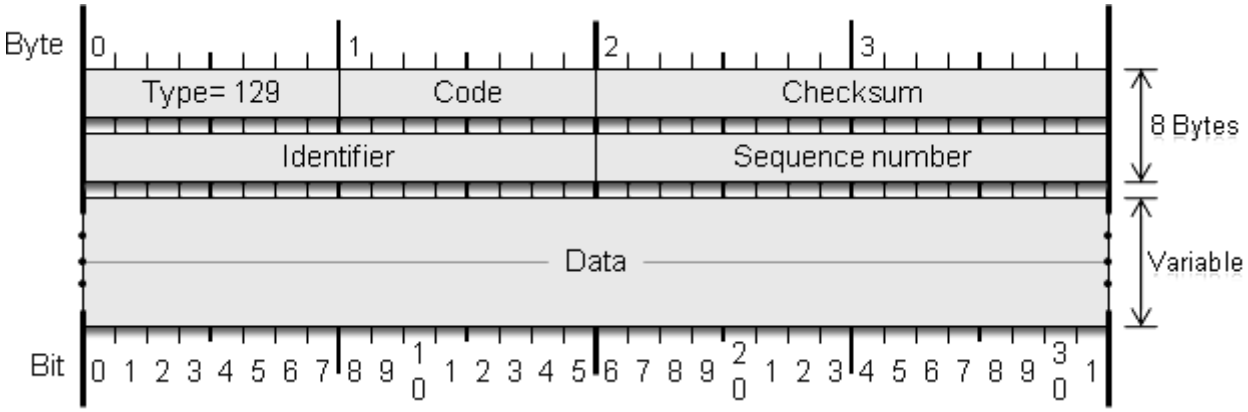

Figura 2.20: Formato del mensaje *Echo Reply*.

2.3 Neighbor Discovery (ND)

La especificación de *Neighbor Discovery* (RFC 4861) presenta diferentes protocolos y procesos ya conocidos en IPv4 que han sido modificados y mejorados y han adquirido nuevas funcionalidades.

Este protocolo resuelve una serie de problemas relacionados con la interacción entre los nodos de un mismo enlace. Define mecanismos para resolver cada uno de los siguientes problemas:

✓ **Descubrimiento de router**
Se refiere a como los hosts localizan a los routers que residen dentro del enlace.

✓ **Descubrimiento de prefijo**
Mecanismo mediante el cual los hosts descubren la serie de prefijos (subredes) que definen cuál destino se encuentra dentro del enlace.

✓ **Descubrimiento de parámetros**
Se refiere a como los nodos aprenden los parámetros de enlace (como el MTU) o de Internet (como el *Hop Limit*) a ser colocados en los paquetes salientes.

✓ **Autoconfiguración de direcciones**
Introduce un mecanismo para permitir a los nodos configurar direcciones para sus interfaces.

✓ **Resolución de dirección**
Similar al protocolo ARP en IPv4, se trata de como los nodos aprenden la dirección de capa de enlace (*Link-Layer Address*) de un vecino, dada únicamente la dirección IPv6 del destino.

✓ **Determinación del próximo salto**
El algoritmo que deduce cual debe ser el próximo nodo a visitar para encontrar la ruta hacia el destino. El próximo salto puede ser un router o el propio destino.

✓ **Detección de vecino inalcanzable**
Con este mecanismo los nodos determinan si un vecino deja de ser alcanzable.

✓ **Detección de dirección duplicada**
Se utiliza para detectar si la dirección que se desea asignar a la interfaz está siendo usada por otro nodo.

✓ **Redireccionamiento**
Utilizado por los routers para informar a un host si existe un mejor primer salto para alcanzar un destino en particular.

El protocolo *Neighbor Discovery* define cinco tipos de paquetes ICMPV6: *Router Solicitation*, *Router Advertisement*, *Neighbor Solicitation*, *Neighbor Advertisement* y el mensaje *Redirect*.

2.3.1 Router Solicitation y Router Advertisement

Cada router de un enlace hace envíos *multicast* periódicamente de paquetes *Router Advertisement* (RA), anunciando su disponibilidad. Un host recibe estos paquetes provenientes de todos los routers, construyendo una lista de routers por defecto.

Los mensajes RA contienen una lista de prefijos usados para determinar si el destino de un paquete se encuentra dentro del enlace o detrás de un router. También son utilizados para la autoconfiguración de las direcciones de los hosts.

Los routers utilizan los mensajes RA para indicarle a los hosts como deben realizar la autoconfiguración de sus direcciones, especificando si debe ser mediante autoconfiguración sin estado (*stateless autoconfiguration*) y/o DHCPv6.

Los mensajes RA también contienen parámetros de Internet como el *Hop Limit* (límite de saltos) y parámetros de enlace como el MTU. Los hosts deben utilizar estos parámetros en los paquetes salientes. Esto facilita la administración centralizada de parámetros críticos que pueden ser indicados en los routers y que son propagados automáticamente a todos los nodos.

Cuando se activa una interfaz, el host envía en *multicast* (FF02::2) un mensaje *Router Solicitation* (RS), el cual solicita a los routers la generación de un mensaje RA en lugar de esperar por el próximo que se tenga programado.

✓ **Router Solicitation**
La cabecera IPv6 de este tipo de mensaje contiene los siguientes valores:

- **Source Address**
  La dirección IPv6 asignada a la interfaz emisora o la dirección no especificada ("::"), si la interfaz no posee una dirección asignada aún.

- **Destination Address**
  La dirección *multicast* de todos los routers (FF02::2).

- **Hop Limit**
  El campo *Hop Limit* posee el valor 255.

El formato del mensaje RS es detallado en la Figura 2.21, extraída de [12].

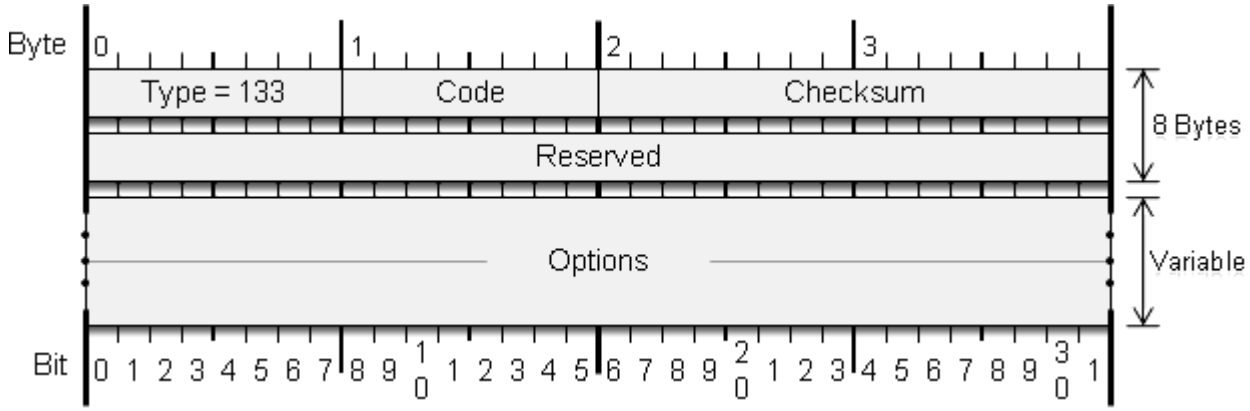

Figura 2.21: Formato del mensaje *Router Solicitation*.

- **Type**
  El tipo de mensaje ICMPv6 de un RS corresponde al valor 133.

- **Code**
  El campo *Code* no es utilizado y posee valor cero.

- **Checksum**
  El campo *Checksum* de ICMPv6 para detectar errores de transmisión.

- **Reserved**
  El campo de 32 bits *Reserved* se encuentra reservado para uso futuro, por lo cual su valor es fijado en cero.

- **Options**
  El campo *Options* puede o no estar presente, la única opción válida hasta el momento para este mensaje es *Source link-layer address*.

✓ **Router Advertisement**

La cabecera IPv6 de este tipo de mensaje contiene los siguientes valores:

- **Source Address**
  Debe ser la dirección link-local de la interfaz emisora.

- **Destination Address**
  La dirección del campo *Source Address* de un mensaje RS, en caso de ser un RA solicitado. La dirección *multicast* de todos los nodos (FF02::1) para un RA no solicitado.

- **Hop Limit**
  El campo *Hop Limit* posee el valor 255.

El formato del mensaje RA es detallado en la Figura 2.22, extraída de [12].

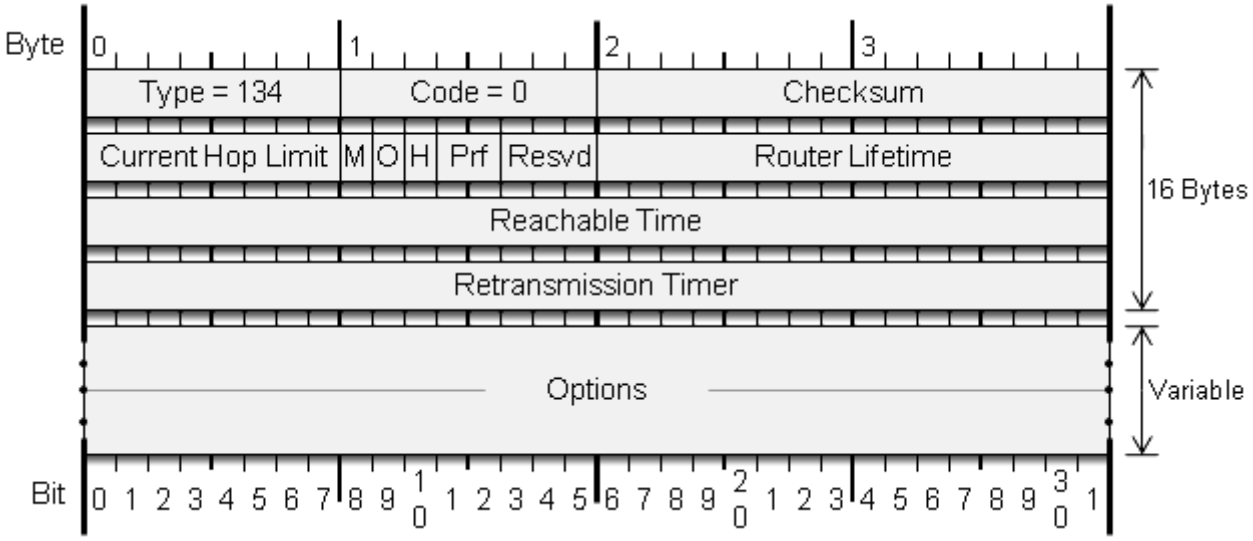

Figura 2.22: Formato del mensaje *Router Advertisement*.

- **Type**
  El tipo de mensaje ICMPv6 de un RA corresponde al valor 134.

- **Code**

El campo *Code* no es utilizado y posee valor cero.

- **Checksum**

El campo *Checksum* de ICMPv6 para detectar errores de transmisión.

- **Current Hop Limit**

El campo de 8 bits *Current Hop Limit* es utilizado para configurar el límite de saltos por defecto de todos los nodos del enlace. Si este campo posee valor cero quiere decir que no es especificado y los hosts mantienen su valor actual.

- **M**

La bandera de 1 bit *Managed Address Configuration* indica la forma en que los nodos configuran su dirección. Si toma valor cero, todos los nodos del enlace usarán únicamente la autoconfiguración sin estado. Si toma valor 1 los nodos configurarán sus direcciones mediante la autoconfiguración sin estado y DHCPv6.

- **O**

La bandera de 1 bit *Other Configuration* con valor 1 indica que los nodos utilizarán DHCPv6 para información adicional (no relacionada con las direcciones).

- **H**

La bandera de 1 bit *Home Agent* posee valor 1, si el router es un *home agent*[4] en el enlace.

- **Prf**

La bandera de 2 bits *Preference* se utiliza para indicar la preferencia o prioridad que tendrá este router sobre otros routers por defecto. Los valores posibles se pueden apreciar en la Tabla 2.7 [13].

- **Resvd**

El campo de 3 bits *Reserved* se encuentra reservado para uso futuro, por lo cual su valor es fijado en cero.

[4] Home Agent es un router de la red “home” el cual gestiona la localización de los dispositivos móviles. Es utilizado en *Mobile IPv6* [14].

Valor	Preferencia
01	Alta
00	Media
11	Baja
10	Reservado (no debe usarse)

Tabla 2.7: Valores de preferencia.

- **Router Lifetime**

El campo de 16 bits *Router Lifetime* es importante sólo si el router será utilizado como router por defecto para los nodos del enlace. Con valor cero indica que no es un router por defecto. Cualquier otro valor especifica la cantidad de segundos que los nodos deberán mantener este router como router por defecto. El valor máximo de tiempo de vida es de 18,2 horas.

- **Reachable Time**

El campo de 32 bits *Reachable Time* indica el tiempo en milisegundos que un nodo asume que un vecino es alcanzable luego de recibir una confirmación de alcance. Si posee valor cero quiere decir que el router no especifica esta información.

- **Retrans Timer**

El campo de 32 bits *Retrans Time* es el tiempo en milisegundos que debe esperar un host entre dos mensajes *Neighbor Solicitation* consecutivos.

- **Options**

El campo de 128 bits *Options* puede contener de cero a varias opciones. Las posibles opciones son:

  - **Source link-layer address**

    Indica la dirección de capa de enlace de la interfaz por la que fue enviado el mensaje RA.

  - **MTU**

    Debe ser enviada en enlaces donde el MTU es variable. Ejemplo: *Token Ring*.

  - **Prefix Information**

    Esta opción indica los prefijos que se encuentran dentro del enlace y/o que son utilizados en la autoconfiguración de direcciones.

- **Route Information**

Establece una preferencia a una ruta accesible por el router. Esto le permite a los hosts escoger el mejor router para el envío de un mensaje.

- **Recursive DNS Server**

Publica direcciones de servidores DNS (Domain Name System) recursivos, permitiendo a los hosts configurar las direcciones de sus servidores DNS.

- **Advertisement Interval**

Anuncia el intervalo de tiempo utilizado por el router para el envío de mensajes RA no solicitados.

- **Home Agent Information**

Anuncia información específica del funcionamiento del router como un *home agent*.

2.3.2 Neighbor Solicitation y Neighbor Advertisement

Los nodos realizan resolución de direcciones mediante envíos *multicast* de mensajes *Neighbor Solicitation* (NS). Este mensaje solicita al nodo destino su dirección link-layer. El NS es enviado a la dirección *multicast solicited-node*. El nodo solicitado responde con su dirección de capa de enlace mediante un mensaje *Neighbor Advertisement* (NA).

Una simple petición y respuesta es suficiente para que ambos nodos conozcan sus direcciones de capa 2 ya que el nodo incluye su dirección de capa de enlace en el mensaje NS.

Un nodo que sabe que su dirección de capa de enlace ha cambiado, puede enviar mediante *multicast* algunos mensajes NA al resto de los nodos. Al recibir los nodos el mensaje el NA, deben actualizar su cache de vecinos con la nueva dirección de capa de enlace [12].

Estos mensajes también son utilizados en los procesos *Duplicate Address Detection* (detección de direcciones duplicadas) y *Neighbor Unreachability Detection* (detección de vecino inalcanzable).

✓ **Neighbor Solicitation**

La cabecera IPv6 de este tipo de mensaje contiene los siguientes valores:

- **Source Address**
Puede tener la dirección perteneciente a la interfaz emisora o la dirección no especificada ("::"), en el caso de aplicarse el proceso de detección de direcciones duplicadas.

- **Destination Address**
Puede tener la dirección asignada al campo *Target Address* o la dirección *multicast solicited-node* asociada a la dirección de dicho campo.

El formato del mensaje NS es detallado en la Figura 2.23, extraída de [12].

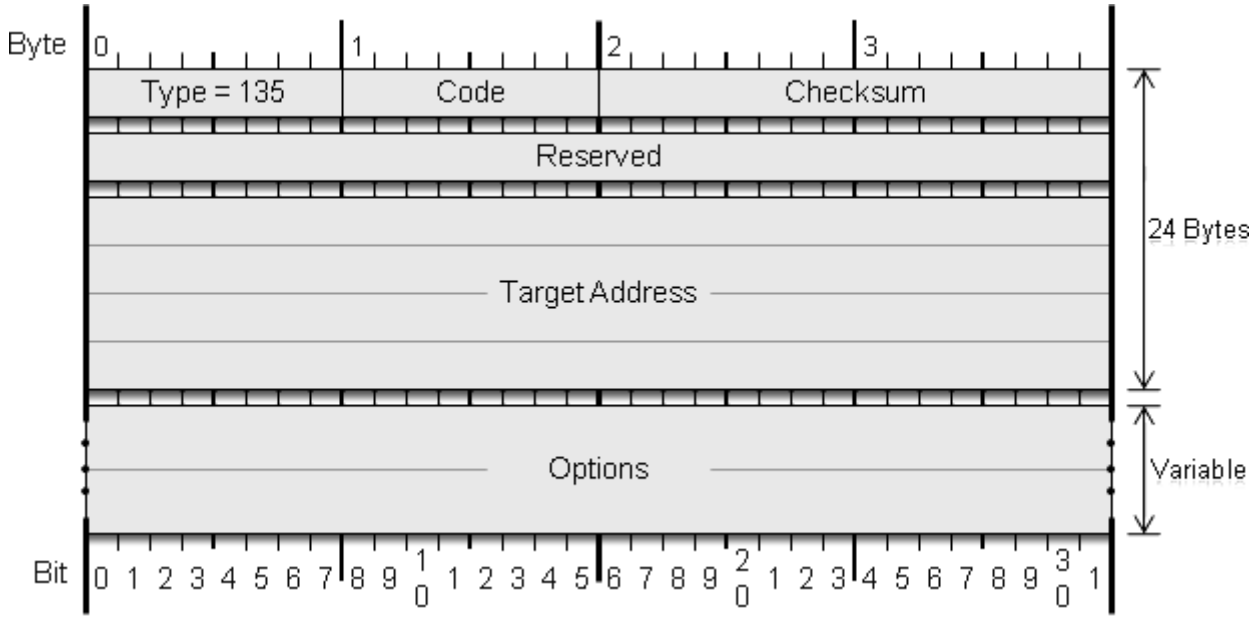

Figura 2.23: Formato del mensaje *Neighbor Solicitation*.

- **Hop Limit**
El campo *Hop Limit* posee el valor 255.

- **Type**
El tipo de mensaje ICMPv6 de un NS corresponde al valor 135.

- **Code**
El campo *Code* no es utilizado y posee valor cero.

- **Checksum**
El campo *Checksum* de ICMPv6 para el manejo de errores de transmisión.

- **Reserved**
El campo de 32 bits *Reserved* se encuentra reservado para uso futuro, por lo cual su valor es fijado en cero.

- **Target Address**
  El campo de 128 bits *Target Address* posee la dirección a la cual se está realizando la solicitud. No puede ser una dirección *multicast*.

- **Options**
  El campo *Options* puede contener la opción *Source link-layer address*. Es usado durante la resolución de dirección de capa red a dirección de capa de enlace y en el proceso detección de vecino inalcanzable. No debe ser incluida si la dirección de origen es la dirección no especificada.

✓ **Neighbor Advertisement**

La cabecera IPv6 de este tipo de mensaje contiene los siguientes valores:

- **Source Address**
  Debe ser la dirección de la interfaz emisora.

- **Destination Address**
  En caso de un mensaje NA solicitado, lleva la dirección del campo *Source Address* del mensaje NS, si esa dirección es la no especificada o es un mensaje NA no solicitado, entonces el valor asignado es la dirección *multicast* de todos los nodos (FF02::1).

- **Hop Limit**
  El campo *Hop Limit* posee el valor 255.

El formato del mensaje NA es detallado en la Figura 2.24, extraída de [12].

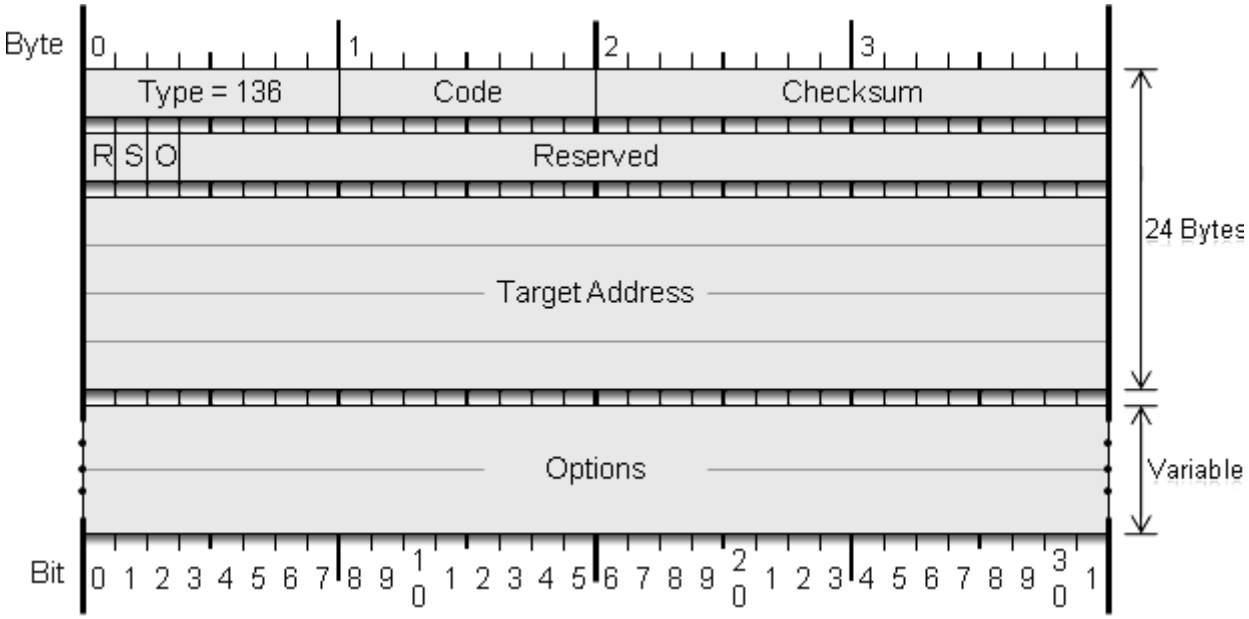

Figura 2.24: Formato del mensaje *Neighbor Advertisement*.

- **Type**

El tipo de mensaje ICMPv6 de un NA corresponde al valor 136.

- **Code**

El campo *Code* no es utilizado y posee valor cero.

- **Checksum**

El campo *Checksum* de ICMPv6 para el manejo de errores de transmisión.

- **R**

La bandera de 1 bit *Router* indica que el emisor es un router.

- **S**

La bandera de 1 bit *Solicited* indica que el mensaje es enviado como respuesta a un mensaje NS.

- **O**

La bandera de 1 bit *Override* indica con valor 1, que la información que viaja en el NA deberá reemplazar la información almacenada en la cache del nodo y actualizar cualquier dirección de capa de enlace. Si su valor es cero, no actualizará ninguna información pero si agregará direcciones de capa 2 que no se encuentren en la cache.

- **Reserved**

El campo de 29 bits *Reserved* se encuentra reservado para uso futuro, por lo cual su valor es fijado en cero.

- **Target Address**

El campo de 128 bits *Target Address* contiene el valor del campo *Target Address* del mensaje NS recibido, en caso de un mensaje NA solicitado. Si no fue solicitado, contiene la dirección de la interfaz cuya dirección de capa de enlace ha cambiado.

- **Options**

El campo *Options* puede contener la opción *Target link-layer address*.

2.3.3 Redirect

Los routers envían paquetes *Redirect* para informar a un host emisor sobre el mejor primer salto en la vía de un destino. También son utilizados para informarle a un host que el destino es en realidad un vecino en el mismo enlace y no un nodo en una subred remota.

La cabecera IPv6 de este tipo de mensaje contiene los siguientes valores:

- **Source Address**

Debe ser la dirección link-local de la interfaz emisora.

- **Destination Address**

La dirección del campo *Source Address* del paquete que activó el mensaje *Redirect*.

- **Hop Limit**

El campo *Hop Limit* posee el valor 255.

El formato del mensaje *Redirect* es detallado en la Figura 2.25, extraída de [12].

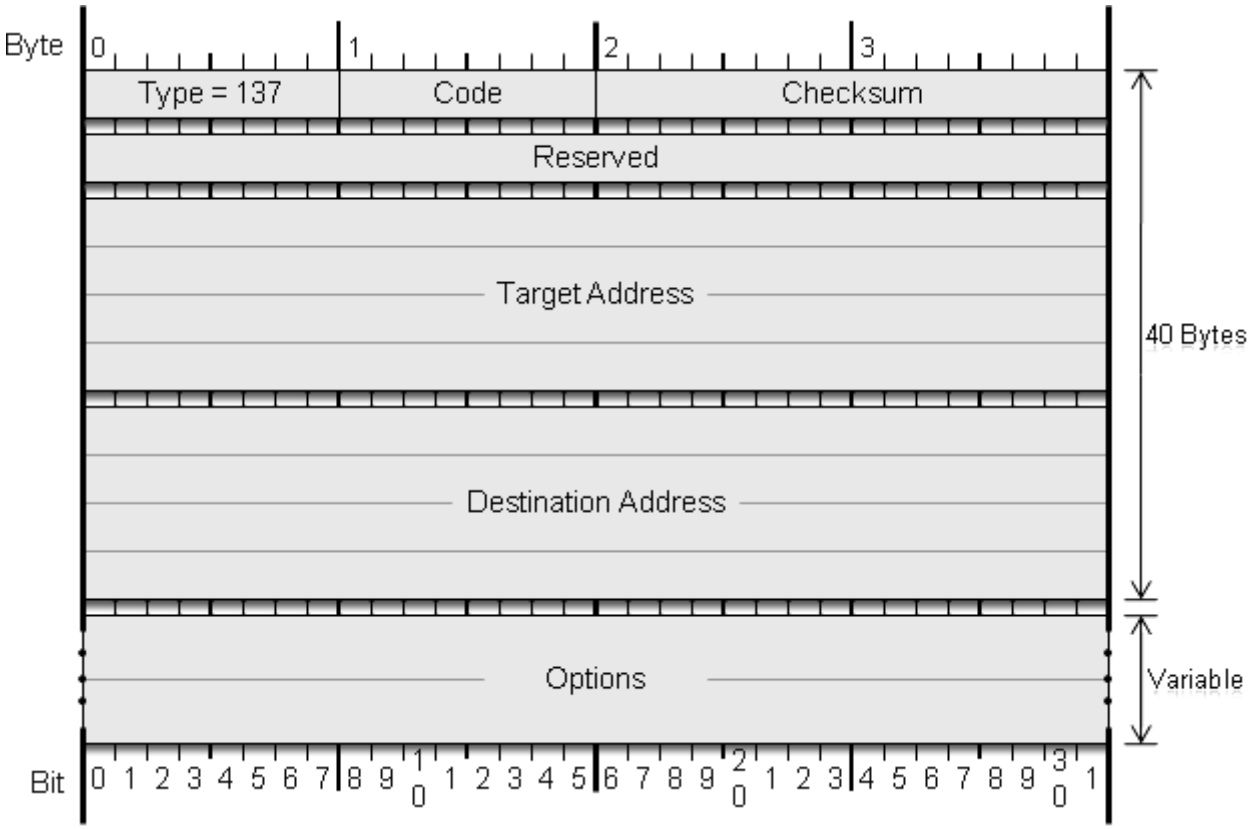

Figura 2.25: Formato del mensaje *Redirect*.

- **Type**

El tipo de mensaje ICMPv6 de un RA corresponde al valor 137.

- **Code**

El campo *Code* no es utilizado y posee valor cero.

- **Checksum**

El campo *Checksum* de ICMPv6.

- **Reserved**

El campo de 32 bits *Reserved* se encuentra reservado para uso futuro, por lo cual su valor es fijado en cero.

- **Target Address**

El campo de 128 bits *Target Address* contiene la dirección link-local de la interfaz que es considerado como mejor primer salto (*first-hop*).

- **Destination Address**

El campo de 128 bits *Destination Address* posee la dirección del destino que está siendo redireccionado.

- **Options**

El campo *Options* puede contener las siguientes opciones:

  - **Target link-layer address**

  La dirección de capa de enlace asociada a la dirección IPv6 del campo *Target Address*.

  - **Redirected Header**

  Contiene toda la información posible de la cabecera IP del paquete que ocasionó el envío del mensaje *Redirect*, sin exceder el MTU mínimo de IPv6, 1280 bytes.

2.3.4 Opciones

Los mensajes del protocolo *Neighbor Discovery* incluyen cero o más opciones, de las cuales, algunas aparecen múltiples veces en un mismo mensaje. Las opciones deben ser rellenadas (*padding*) con los bits necesarios para completar una cantidad múltiplo de 64 bits. Todas las opciones poseen el formato mostrado en la Figura 2.26 (tomada de [12]):

- **Type**

El campo de 8 bits *Type* indica el tipo de opción.

- **Length**

El campo de 8 bits *Length* especifica la longitud de la opción incluyendo todos los campos. Es expresada en unidades de 8 bytes (64 bits). Cualquier opción cuya longitud sea cero deberá ser ignorada.

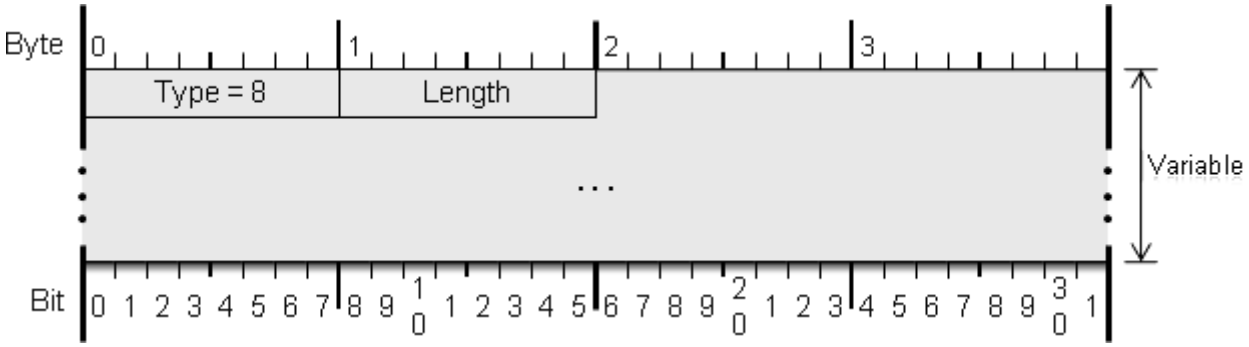

Figura 2.26: Formato de las opciones *Neighbor Discovery*.

✓ **Source/Target Link-Layer Address**

La opción *Source Link-Layer Address* contiene la dirección de capa de enlace del emisor del paquete. Es utilizada en los mensajes NS, RS y RA

La opción Target Link-Layer Address es utilizada por los mensajes NA y *Redirect*. Contiene la dirección de capa de enlace asociada a la dirección del campo *Target Address*.

La Figura 2.27 (tomada de [12]) muestra el formato de ambas opciones.

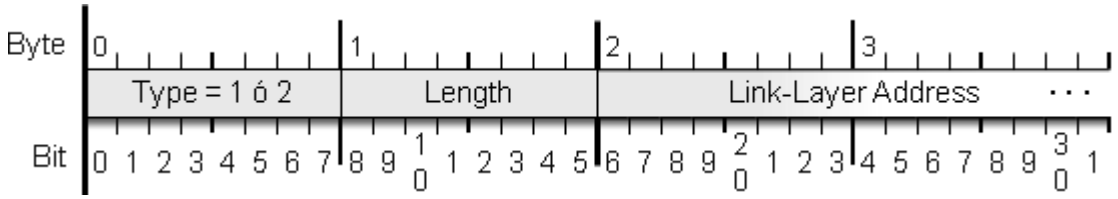

Figura 2.27: Formato de la opción *Source/Target Link-Layer Address*.

- **Type**

  Puede tomar dos valores, 1 para la opción *Source Link-Layer Address* y 2 para la opción *Target Link-Layer Address*.

- **Length**

  La longitud depende de la dirección de capa de enlace, para el caso de la dirección MAC, la longitud tomaría valor uno 1, indicando 8 bytes.

- **Link-Layer Address**

  El campo de tamaño variable *Link-Layer Address* contiene la dirección de capa de enlace del destino o del origen.

✓ **Prefix Information**

Provee a los hosts con prefijos que se encuentran dentro del enlace y prefijos para la autoconfiguración de direcciones. La opción *Prefix Information* es

utilizada únicamente por los mensajes RA. El formato de esta opción se muestra en la Figura 2.28 (tomada de [12]).

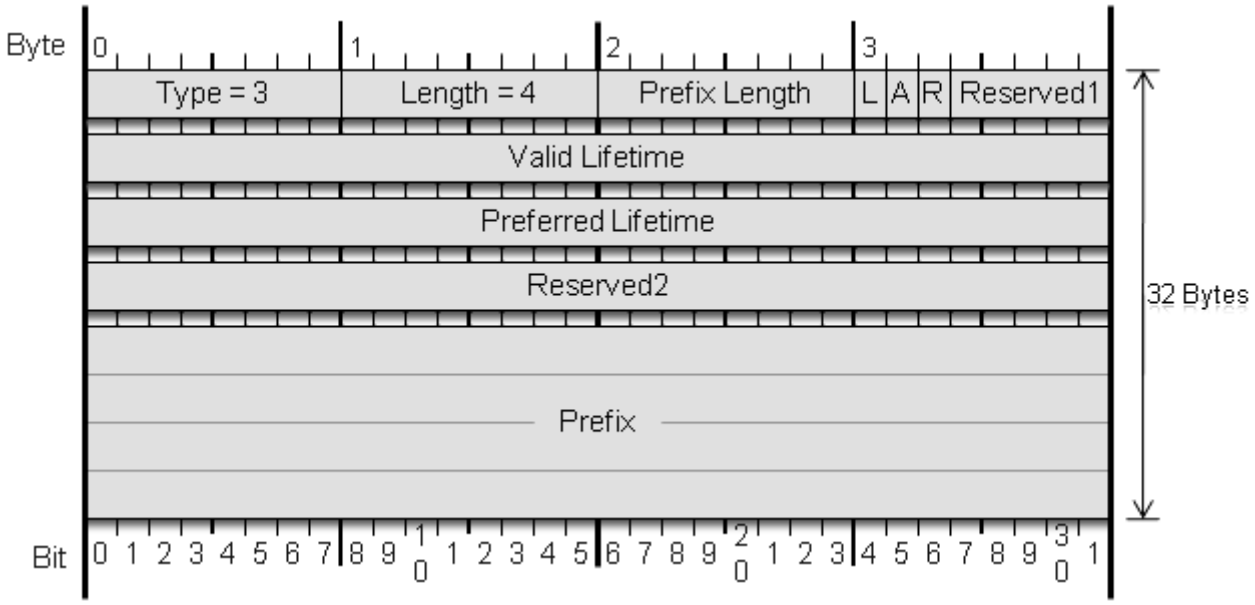

Figura 2.28: Formato de la opción *Prefix Information*.

- **Type**

El campo *Type* posee el valor 3.

- **Length**

El campo *Length* debe ser fijado en 4, lo que quiere decir que el encabezado posee 32 bytes.

- **Prefix Length**

El campo de 8 bits *Prefix Length* indica la longitud del prefijo.

- **L**

La bandera de 1 bit *On-Link* posee valor 1 si el prefijo se encuentra dentro del enlace. Si su valor es cero, no se posee ninguna información al respecto.

- **A**

La bandera de 1 bit *Autonomous address-configuration* indica con valor 1 que el prefijo puede ser utilizado para configuración de direcciones.

- **R**

La bandera de 1 bit *Router Address* indica con valor 1 que el campo *Prefix* contiene una dirección IPv6 completa perteneciente al router emisor [14].

- **Reserved1**

El campo de 5 bits *Reserved1* se encuentra reservado para uso futuro, por lo cual su valor es fijado en cero.

- **Valid Lifetime**

El campo de 32 bits *Valid Lifetime* indica el tiempo en segundos que el prefijo debe ser válido para propósitos de determinaciones de rutas en el enlace. Todos los bit en 1 significa tiempo infinito.

- **Preferred Lifetime**

El campo de 32 bits *Preferred Lifetime* es la cantidad de segundos que las direcciones generadas con el prefijo (por medio de la *Stateless Address Autoconfiguration*) permanecen en estado *preferred*. El valor de *Preferred Lifetime* no debe exceder al *Valid Lifetime*.

- **Reserved2**

El campo de 32 bits *Reserved2* se encuentra reservado para uso futuro, por lo cual su valor es fijado en cero.

- **Prefix**

El campo de 128 bits *Prefix* contiene una dirección IPv6 o un prefijo. El resto de bits del campo no pertenecientes al prefijo deben contener valor cero.

✓ **Redirected Header**

Es utilizada en los mensajes *Redirect*, contienen todo o parte del paquete que está siendo redireccionado. La Figura 2.29 (tomada de [12]) muestra el formato de la cabecera.

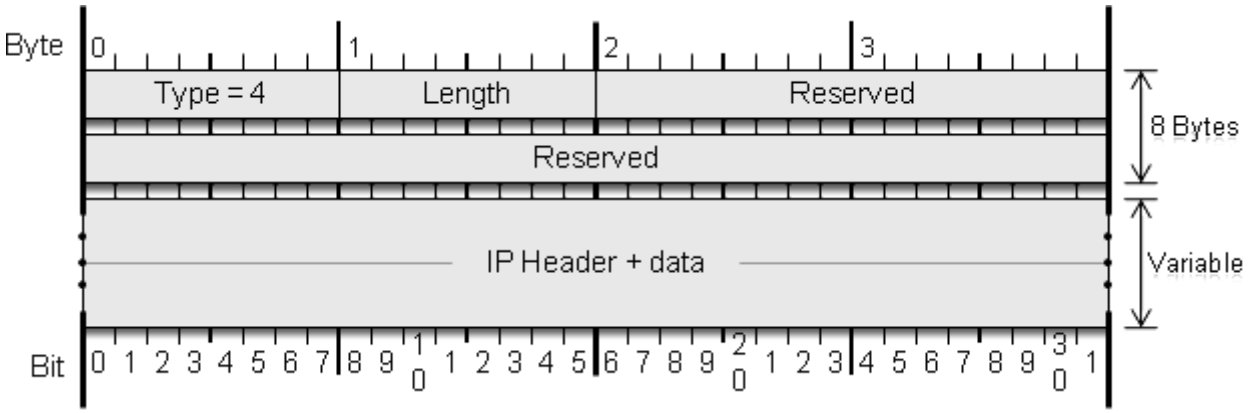

Figura 2.29: Formato de la opción *Redirected Header*.

- **Type**

El campo *Type* posee el valor 4.

- **Length**

La longitud es variable.

- **Reserved**

El campo de 48 bits *Reserved* se encuentra reservado para uso futuro, por lo cual su valor es fijado en cero.

- **IP Header + data**

El campo *IP Header + data* contiene el paquete que será redireccionado, truncado de manera que no se exceda el MTU.

✓ **MTU**

La opción *MTU* es utilizada en los mensajes RA para asegurar que todos los nodos en el enlace usen el mismo MTU. En la Figura 2.30 (tomada de [12]) se muestra el formato de la opción *MTU*.

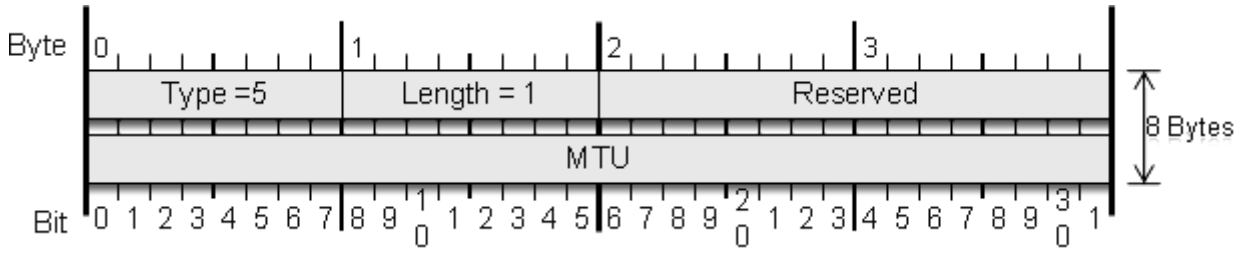

Figura 2.30: Formato de la opción *MTU*.

- **Type**

El campo *Type* posee el valor 5.

- **Length**

El campo *Length* posee el valor 1, indicando que es de tamaño 8 bytes.

- **Reserved**

El campo de 18 bits *Reserved* se encuentra reservado para uso futuro, por lo cual su valor es fijado en cero.

- **MTU**

El campo de 32 bits *MTU* posee el MTU recomendado para el enlace.

✓ **Route Information**

Especifica prefijos que son alcanzables por vía del router y les asigna una preferencia. Sólo es utilizada por los mensajes RA. Su formato se muestra en la Figura 2.31, extraída de [12].

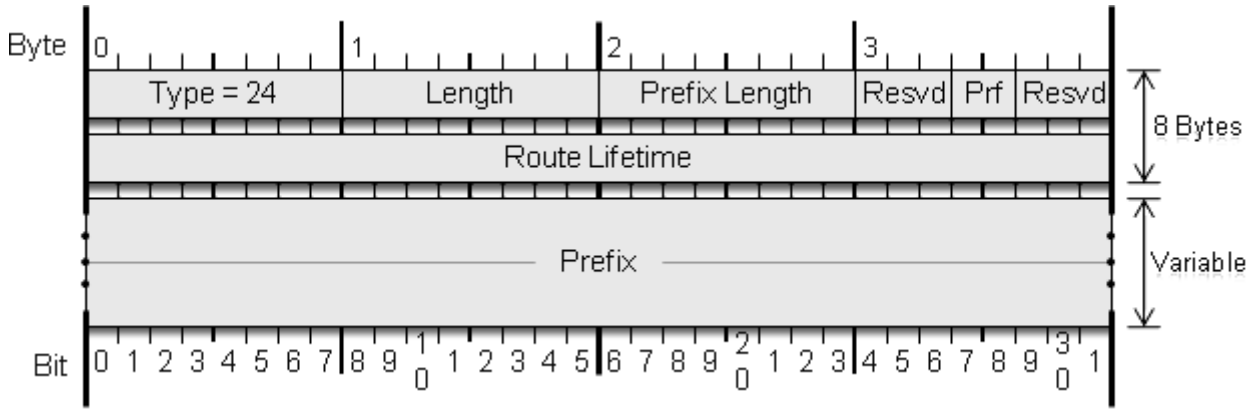

Figura 2.31: Formato de la opción *Route Information*.

- **Type**

El campo *Type* posee el valor 24.

- **Length**

La longitud es variable, depende del prefijo.

- **Prefix Length**

El campo de 8 bits *Prefix Length* indica la cantidad de bits del campo *Prefix*, puede presentar los valores 0, 64 o 128 dependiendo del prefijo.

- **Resvd**

El campo de 3 bits *Resvd* se encuentra reservado para uso futuro, por lo cual su valor es fijado en cero.

- **Prf**

La bandera de 2 bits *Preference* se utiliza para indicar la preferencia de la ruta, de igual forma que lo hace el mensaje RA, los posibles valores están en la Tabla 2.7.

- **Resvd**

El campo de 3 bits *Resvd* se encuentra reservado para uso futuro, por lo cual su valor es fijado en cero.

- **Route Lifetime**

El campo de 32 bits *Route Lifetime* indica el tiempo en segundos que se debe considerar el prefijo como una ruta válida a través del router. Los 32 bits con valor 1 indican tiempo infinito.

- **Prefix**

El campo *Prefix* posee una longitud variable. Debe ser rellenado con bits en cero para completar una longitud múltiple de 64 bits.

✓ **Recurvise DNS Server**

Utilizada en los mensajes RA, informa a los hosts sobre direcciones e servidores DNS recursivos. Contiene una o más direcciones IPv6 de servidores DNS, todas las direcciones comparten el mismo tiempo de vida (Lifetime). Pueden ser utilizadas múltiples opciones RDNSS en un único mensaje RA. Su formato se muestra en la Figura 2.32 extraída de [15].

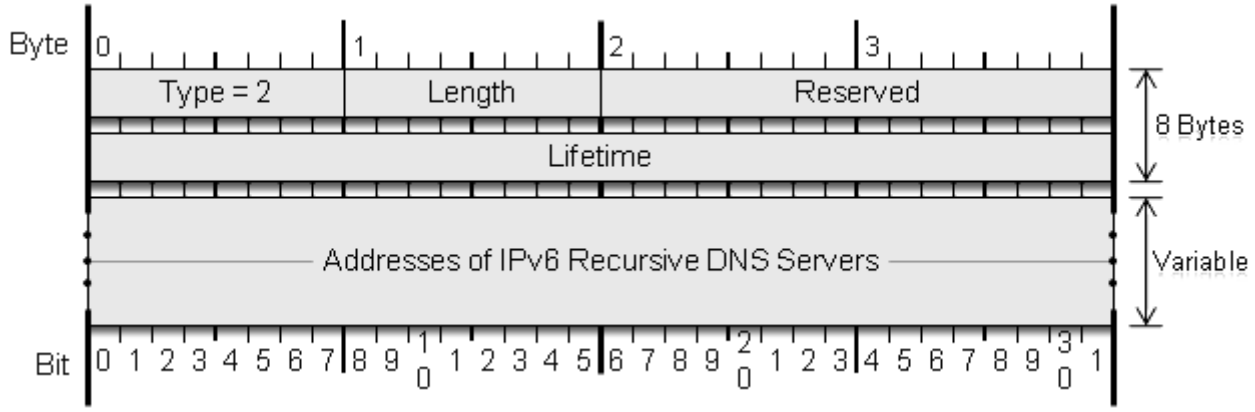

Figura 2.32: Formato de la opción *Recurvise DNS Server*.

- **Type**

El campo *Type* posee el valor 25.

- **Length**

La longitud es variable, depende de la cantidad de direcciones. Es usado por el receptor para determinar la cantidad de direcciones IPv6 en la opción.

- **Lifetime**

El campo de 32 bits *Lifetime* indica el tiempo máximo en segundos, en cual la dirección del servidor DNS será utilizada para resoluciones de nombres. Los 32 bits con valor 1 indican tiempo infinito.

- **Addresses of IPv6 Recursive DNS Servers**

  Una o más direcciones IPv6 de los servidores DNS recursivos.

✓ **Advertisement Interval**

El protocolo Mobile IPv6 define la opción *Advertisement Interval*, usada en los mensajes RA para anunciar el intervalo de tiempo utilizado por el router para el envío de mensajes RA no solicitados. El formato del mensaje se muestra en la Figura 2.33, tomada de [14].

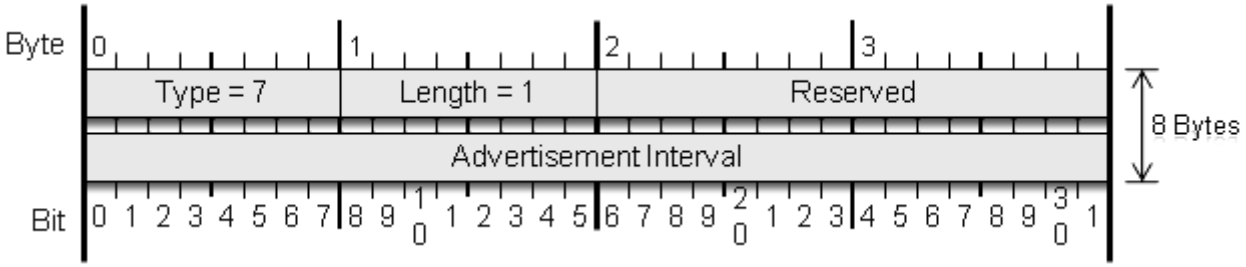

Figura 2.33: Formato de la opción *Advertisement Interval*.

- **Type**

  El campo *Type* posee el valor 7.

- **Length**

  El campo *Length* posee el valor 1, indicando que es de tamaño 8 bytes.

- **Reserved**

  El campo de 18 bits *Reserved* no es usado, su valor debe ser fijado en cero.

- **Advertisement Interval**

  El campo de 32 bits *Advertisement Interval* indica el tiempo máximo en milisegundos, entre cada envío de mensajes RA no solicitados hechos por el router.

✓ **Home Agent Information**

Incluida en los mensajes RA para anunciar información específica del funcionamiento del router como un *home agent*. El formato del mensaje se muestra en la Figura 2.34, tomada de [14] y [16].

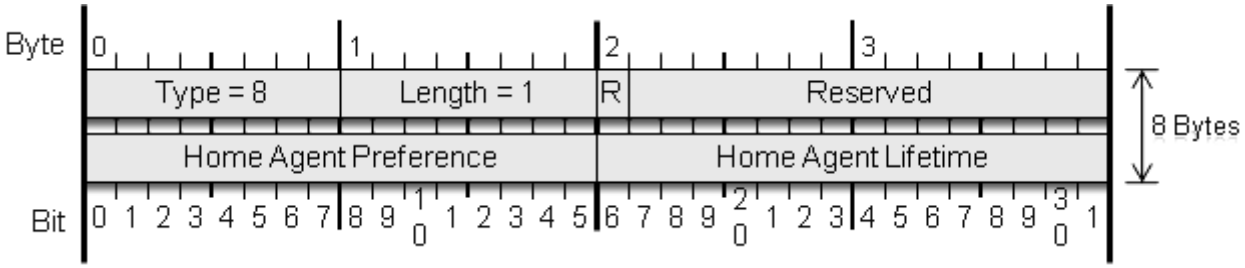

Figura 2.34: Formato de la opción *Home Agent Information*.

- **Type**

El campo *Type* posee el valor 8.

- **Length**

El campo *Length* posee el valor 1, indicando que es de tamaño 8 bytes.

- **R**

La bandera de 1 bit *R* indica con valor 1 que el *home agent* posee soporte de movilidad de routers.

- **Reserved**

El campo de 17 bits *Reserved* se encuentra reservado para uso futuro, su valor debe ser fijado en cero.

- **Home Agent Preference**

El campo de 16 bits *Home Agent Preference* indica la preferencia del *home agent* emisor del mensaje RA. Un valor más alto indica mayor preferencia sobre otro *home agent* con menor valor.

- **Home Agent Lifetime**

El campo de 16 bits *Home Agent Lifetime* indica el tiempo en segundos en que el router ofrecerá servicios de *home agent*.

2.3.5 Procesos de Neighbor Discovery

✓ **Neighbor Unreachability Detection**

Un vecino es considerado alcanzable si el nodo ha recibido recientemente una confirmación de que el paquete que envió a su vecino ha sido entregado. Esta confirmación puede ser de dos formas: un mensaje NA en respuesta a un mensaje NS o un proceso de capa de transporte que indica que la conexión ha sido establecida. En este caso el recibo de mensajes TCP indican que el vecino es alcanzable.

Para mantenerse informado de las conexiones que se encuentran activas y alcanzables, los nodos IPv6 usan diferentes tablas. Dos tablas importantes relacionadas al protocolo *Neighbor Discovery* son *Neighbor Cache* y *Destination Cache*.

- **Neighbor Cache**

  La cache de vecinos mantiene una lista de los vecinos con los que ha existido alguna comunicación. Los vecinos están listados por su dirección *unicast* y cada entrada contiene información de la dirección de capa de enlace y de una bandera que indica si el vecino es un router o un host. Esta tabla puede ser comparada con la cache ARP de IPv4. Las entradas también contienen información sobre el alcance de los vecinos (estado, véase la Tabla 2.8 extraída de [3]) y el tiempo que se debe esperar antes del nuevo chequeo de alcance.

Estado	Descripción
Incomplete	La resolución de dirección se encuentra en espera de una respuesta o a que se acabe el tiempo de espera.
Reachable	Este vecino es actualmente alcanzable y significa que recibió una confirmación de alcance dentro del tiempo de espera.
Stale	Se ha acabado el tiempo de espera desde la última confirmación recibida.
Delay	Cuando el alcance del vecino ha expirado y no se ha recibido una confirmación en el tiempo estipulado, se envía un NS y se cambia el estado a Probe.
Probe	Una confirmación de alcance es solicitada enviando un NS periódicamente hasta que sea confirmado el alcance [3].

Tabla 2.8: Estados de la cache de vecinos (*Neighbor Cache*).

- **Destination Cache**

  Esta tabla también contiene información acerca de los destinos con los que ha habido comunicación recientemente, incluyendo destinos locales y remotos. En caso de destinos remotos se lista la dirección de capa de enlace del siguiente salto. La cache de destino se actualiza con la información recibida en los mensajes ICMPv6 *Redirect*. También puede contener información acerca de los tamaños de MTU.

✓ **Address Resolution**

Es el proceso por el cual un nodo determina la dirección de capa de enlace de un vecino, sabiendo únicamente su dirección IPv6. Una resolución de dirección nunca es realizada sobre direcciones *multicast*.

- **Inicialización de Interfaz**

Cuando se habilita una interfaz, el nodo debe unirse al grupo *multicast* de todos los nodos del enlace, así como también a la dirección *solicited-node multicast* correspondiente a cada una de las direcciones asignadas a la interfaz.

- **Enviando Neighbor Solicitation**

Cuando un nodo desea enviar un paquete *unicast* a un vecino pero no conoce su dirección de capa de enlace, realiza una resolución de dirección. Un mensaje NS es enviado a la dirección *multicast solicited-node* correspondiente a la dirección destino. En la cache de vecinos se crea una entrada con la dirección a resolver y se coloca en estado incompleto.
El emisor debe incluir su dirección de capa de enlace en el mensaje NS por medio de la opción *Source Link-Layer*, de esta forma el receptor podrá responder con el mensaje NA correspondiente.

Mientras se espera la respuesta a la solicitud, el emisor debe reenviar otros mensajes NS en un intervalo regular de tiempo, aun en la ausencia de tráfico adicional.

Si no se recibe ningún mensaje NA luego de varias solicitudes, la resolución de dirección ha fallado. El emisor debe retornar un mensaje ICMPv6 *Destination Unreachable* con código 3 (*Address Unreachable*) para cada paquete que está esperando por la resolución de dirección [12].

- **Recibiendo Neighbor Solicitation**

El receptor del mensaje NS crea o actualiza en la cache de vecinos la entrada correspondiente a la dirección IPv6 origen de la solicitud.

Luego de realizar alguna actualización en la cache de vecinos, el nodo envía un mensaje NA.

- **Enviando Neighbor Advertisement Solicitado**

Un nodo envía un mensaje NA en respuesta a un mensaje NS. El campo *Target Address* del mensaje NS es copiado en el mensaje NA, si el nodo es un router, se coloca valor uno a la bandera *Router Flag*, en caso contrario la bandera permanece con valor cero.

- **Recibiendo Neighbor Advertisement**

Cuando se recibe un mensaje NA válido, se busca en la cache de vecinos la entrada de la dirección a resolver. Si no existe, el advertisement debe ser descartado. En caso contrario, se actualiza la entrada con la dirección

de capa de enlace que viene como una opción en el mensaje NA y se coloca en estado alcanzable [12].

✓ **Duplicate Address Detection (DAD)**

La detección de direcciones duplicadas se debe realizar en todas las direcciones *unicast* antes de que sean asignadas a una interfaz, independientemente si fueron obtenidas por medio de la autoconfiguración sin estado, por DHCPv6 o por configuración manual. DAD no debe ser aplicado en direcciones *anycast* puesto que el propósito de las mismas es dar una misma dirección IPv6 a varias interfaces.

El proceso para detectar direcciones duplicadas utiliza los mensajes NS y NA. Si una dirección duplicada es detectada durante el proceso, no puede ser asignada a la interfaz. Si la dirección es derivada del identificador de una interfaz entonces se le debe asignar un nuevo identificador, o configurar las direcciones manualmente [17].

Una dirección en la que se aplica el proceso DAD, se dice que es "tentativa" hasta que el proceso se complete satisfactoriamente. Esto es logrado uniéndose al grupo *multicast solicited-node* asociado a la dirección en cuestión y enviando uno o más mensajes NS para la dirección. Posteriormente ocurre uno de los siguientes casos:

- El host recibe un mensaje NA de otro host que posee esa dirección.
- El host recibe un mensaje NS de otro host que está realizando DAD.
- No hay respuesta.

Los primeros dos casos indican un conflicto y la dirección es etiquetada como "duplicada". Cuando esto ocurre la dirección no es utilizada. Sólo cuando no hay respuesta es utilizada la dirección. Si hay algún conflicto el sistema debe esperar por una configuración manual [18].

✓ **Redirect Function**

Los routers utilizan la función de redireccionamiento para informar a los hosts emisores sobre un mejor vecino primer salto, por el cual el tráfico debería ser retransmitido hasta un destino en específico. Hay dos escenarios donde el redireccionamiento es utilizado:

1. Un router informa al host emisor sobre la dirección IPv6 de un router disponible en el enlace local, el cual se encuentra más cerca del destino. Esta condición puede ocurrir cuando se encuentran múltiples routers en un segmento de red y el host emisor escoge su router por defecto y este no es la mejor vía para alcanzar al destino.

2. Un router le informa al host emisor que el destino es un vecino (se encuentra dentro del mismo enlace del emisor). Esta condición ocurre cuando la lista de prefijos de un host no contiene el prefijo del destino. Debido a que el prefijo del destino no es encontrado en la lista de prefijos, el host emisor envía el paquete hacia tu router por defecto

Los mensajes *Redirect* son enviados únicamente por el primer router en el camino entre el emisor y el destino. Los hosts nunca envían mensajes *Redirect* y los routers nunca actualizan su tabla de enrutamiento basándose en los mensajes *Redirect* recibidos.

2.4 Autoconfiguración de Direcciones

Uno de los aspectos más útiles de IPv6 es su capacidad de configurarse automáticamente, aun sin el uso de un protocolo de autoconfiguración tal como DHCPv6. Un host IPv6 puede autoconfigurar una dirección link-local para cada interfaz [19].

2.4.1 Tipos de Autoconfiguración

Existen tres tipos de autoconfiguración:

- ✓ **Stateless o sin estado**

Las direcciones y demás parámetros son generados en los hosts automáticamente con los mensajes RA enviados por los servidores. Dado que los hosts generan la configuración, esta no es guardada en los servidores, es decir que los servidores no conocen el estado de los hosts del enlace. Los mensajes RA poseen las banderas *Managed Address Configuration* y *Other Configuration* con valor cero e incluyen una o más opciones *Prefix Information* con la bandera *Autonomous* en valor 1.

- ✓ **Stateful**

Configuración basada en el uso de un protocolo de configuración de direcciones, tal como DHCPv6, para obtener direcciones y otros parámetros de configuración. Un host usa autoconfiguración *stateful* cuando recibe un mensaje RA con alguna de las banderas *Managed Address Configuration* y *Other Configuration* con valor 1.

- ✓ **Mixto**

Esta configuración se basa en el recibo de mensajes RA que incluyen información de prefijos, cada uno con la bandera *Autonomous* con valor 1 y además posee alguna de las banderas *Managed Address Configuration* y *Other Configuration* con valor 1.

2.4.2 Ciclo de Vida de una Dirección IPv6 Autoconfigurada

Completado el proceso de autoconfiguración sin estado, la dirección asignada a la interfaz puede ser utilizada hasta que expire el tiempo de vida preferente (*Preferred Lifetime*) indicado en el mensaje RA. En la mayoría de los casos esto no ocurre dado que nuevos mensajes RA recibidos reiniciaran los temporizadores del tiempo de vida. Sin embargo, si no hay más mensajes RA, el tiempo de vida expira y la dirección se convierte en obsoleta (*Deprecated*). En nuevas sesiones no se debe usar las direcciones obsoletas, sino que se debe obtener una dirección preferente si hay alguna disponible. Aun así es posible que en sesiones ya existentes continúe con el uso de alguna dirección obsoleta. Eventualmente, el tiempo de vida válido (*Valid Lifetime*) también se agotará y se removerá la dirección obsoleta de dicha interfaz. Con esto se cerrará cualquier sesión que tenga en uso esa dirección. En la Figura 2.35 (tomada de [4]) se describe el ciclo de vida.

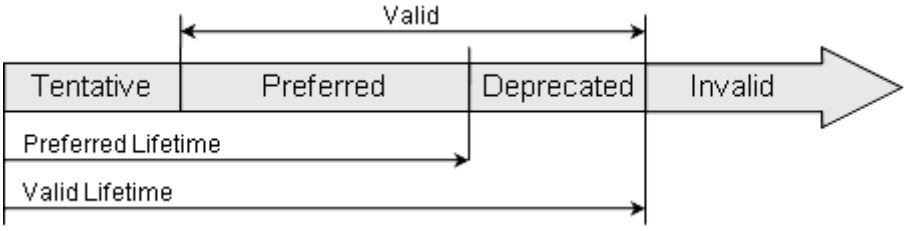

Figura 2.35: Ciclo de vida de una dirección IPv6 autoconfigurada.

2.4.3 Proceso de Autoconfiguración sin Estado

La autoconfiguración sin estado (*stateless address autoconfiguration*) define los procesos necesarios para que un host configure las direcciones de sus interfaces basándose en información disponible localmente y la información distribuida por los routers. Este proceso requiere una mínima configuración en los routers, no requiere de ningún otro tipo de servidor y lo más importante es que no requiere ningún tipo de configuración manual en los hosts.

Cuando un host levanta por primera vez una interfaz ocurren los siguientes pasos:

- El host elige el identificador de una interfaz, que se deriva usualmente de la dirección de capa de enlace.
- Se crea una dirección link-local con prefijo FE80::/64 y el identificador de la interfaz.
- Se verifica que ningún otro nodo esté utilizando la dirección mediante el algoritmo de detección de dirección duplicada.
- El host envía un mensaje RS a todos los routers del enlace para obtener nuevos prefijos. Los routers responden con un mensaje RA.

Estos mensajes RA contienen una lista de prefijos del enlace y un indicador en caso de que el router provea servicios de enrutamiento.

- Por cada prefijo recibido, el host configura una nueva dirección utilizando el identificador de interfaz. Verificando posteriormente su unicidad con el algoritmo detección de dirección duplicada.
- El host se mantiene a la espera de actualizaciones sobre los prefijos del enlace y reconfigura sus direcciones en caso de que sea necesario. Los routers envían estas actualizaciones en mensajes RA no solicitados, bien sean periódicamente o cuando ocurran cambios en la configuración de la red.

La autoconfiguración sin estado no sólo provee información de los prefijos, también le indica a los hosts cual router del enlace está disponible y dispuesto a proveer servicios de enrutamiento. A diferencia de los hosts IPv4, los hosts IPv6 no tendrán un sólo router por defecto, sino una lista de routers por defecto (*default routers*). Cada vez que se trata de enviar un paquete a una dirección fuera del enlace, el host enviará el paquete a uno de los routers y dejará que ese router se encargue del paquete.

Los routers no utilizan la autoconfiguración sin estado para obtener sus direcciones; ellos necesitan ser configurados explícitamente con direcciones e información de los prefijos que deben anunciar [9].

2.4.4 DHCPv6

DHCPv6 está definido en el RFC 3315 para proveer configuración de direcciones *stateful* o (direcciones adicionales) o parámetros para hosts IPv6. Un host usa un protocolo de configuración como DHCPv6, basándose en las siguientes banderas de un mensaje RA enviado por un router vecino:

✓ **Managed Address Configuration**

También conocida como la bandera *M*. Cuando es activada (posee valor 1), le indica al host que debe usar un protocolo de configuración con estado (DHCPv6) para obtener direcciones.

✓ **Other Configuration**

También conocida como la bandera *O*. Cuando se encuentra activada, indica al host que debe usar el protocolo DHCPv6 para obtener otros parámetros de configuración.

La combinación de los valores de las banderas *M* y *O* son las siguientes:

- **M = 0 y O = 0**

Esta combinación corresponde a una red sin soporte para DHCPv6. Los hosts usan la información de los RA para autoconfiguración *stateless*.

- **M = 1 y O = 1**

En esta combinación, DHCPv6 es utilizado para la obtención direcciones y otros parámetros de configuración. Esta combinación es conocida como *DHCP stateful*, en la cual un servidor DHCPv6 asigna direcciones a los hosts IPv6.

- **M = 0 y O = 1**

En esta combinación, DHCPv6 es utilizado únicamente para otros parámetros de configuración. Para las direcciones, se utiliza la autoconfiguración *stateless*. Esta combinación es conocida como *DHCPv6 stateless*.

- **M = 1 y O = 0**

En esta combinación, DHCPv6 es usado para la configuración de direcciones pero no para otros parámetros de configuración. Debido a que un host IPv6 normalmente necesita ser configurado con otros parámetros, tales como la dirección IPv6 del servidor DNS, esta es una combinación poco común.

2.4.5 Delegación de Prefijos

El RFC 3633 describe opciones del protocolo DHCPv6, el cual provee un mecanismo para la delegación de prefijos de IPv6. Mediante estas opciones un router delegante (*delegating router*) le indica a los routers solicitantes (*requesting routers*) los prefijos que deben ser anunciados.

La delegación de prefijos con DHCPv6 es independiente de la asignación de dirección con DHCPv6. Un router solicitante puede usar DHCPv6 sólo para delegación de prefijos o para delegación de prefijos junto con asignación de direcciones y otros parámetros de configuración [20].

3. Marco Metodológico

Una metodología de desarrollo define un método para llevar a cabo el desarrollo de software en forma rápida y eficaz. Cada método posee principios y características que los hace únicos. En este apartado se especifica la metodología utilizada para realizar la aplicación y todos los aspectos tomados en cuenta durante el proceso de desarrollo.

3.1 Metodología de desarrollo *Scrum*

La metodología *Scrum* propone un conjunto de pasos para organizar el desarrollo de proyectos con grupos de trabajo pequeños y con iteraciones de períodos de tiempo relativamente cortos.

Scrum propone cuatro tipos de actores *Product Owner*, *Scrum Master*, *Scrum Team* y cliente o usuario [21]. Para efectos de esta implementación, el *Scrum Team* (programadores) está compuesto a su vez por el *Product Owner* y *Scrum Master* que se encargan de la administración de la aplicación.

Cada actor del proceso posee funciones particulares que ejecuta durante alguna de las etapas del desarrollo. Las etapas que toman lugar durante esta metodología son: *Product Backlog*, *Sprints* y *Daily Scrum Meeting*.

Al completar todos los *Sprints* planificados durante la reunión del *Product Backlog*, se da por finalizada la realización de la aplicación y se hace una prueba final para verificar el completo funcionamiento del programa. Las etapas de la implementación se explican a continuación.

3.1.1 Product Backlog

Durante el *Product Backlog* se establecen las prioridades del proyecto, de forma que se realicen las actividades más importantes primero y posteriormente los complementos. Se definen los *Sprints* o ciclos que se realizarán de acuerdo a selección de prioridades previamente realizada. Se guía al *Scrum Team* con lineamientos para que el trabajo sea uniforme independientemente del individuo.

3.1.2 Sprints

Un *Sprint* está dividido en cuatro etapas:

- ✓ *Sprint Planning Meeting*: al iniciar cada *Sprint* se realiza una reunión donde se levantan los requerimientos necesarios para llevar a cabo el módulo, se definen las metas del *Sprint* (*Sprint Goal*) y se estipula un tiempo de una a cuatro semanas para su realización.

- ✓ *Sprint Backlog*: es el tiempo en el que se lleva a cabo el módulo. Dura entre dos y cuatro semanas y en cada día se realiza un *Scrum Meeting*. Durante el *Sprint Backlog* no se realiza ninguna modificación en la aplicación fuera de lo estipulado en la *Sprint Planning Meeting*.

- ✓ *Sprint Review*: al culminar el *Sprint Backlog* se revisa el módulo realizado. Se explican los procesos realizados a los miembros del equipo y se analiza si hay algún problema que se deba resolver.

- ✓ *Sprint Retrospective*: durante esta fase del *Sprint* se toman en cuenta todos los procesos que no se pudieron culminar o aquellos que durante el *Sprint Review* muestran deficiencias. Culminado el *Sprint Retrospective* se añaden al siguiente *Sprint* todas las nuevas actividades pendientes. Para efectos de este documento en cada *Sprint* se explicará la versión final de cada módulo para mantener el orden lógico de la aplicación y en el apartado *Sprint Retrospective* se indicarán los cambios realizados para dicho *Sprint*.

3.1.3 Scrum

Cada día del *Sprint* los miembros del *Scrum Team* hacen una reunión usualmente presencial para discutir que se hizo el día anterior, que se hará durante el día y que ayudas se podrían necesitar. Esta reunión diaria es denominada *Scrum*. Cuando un miembro del *Scrum Team* prevé que puede haber algún contratiempo se ofrece la ayuda y se realiza la investigación necesaria para solventar el problema.

4. Marco Aplicativo

En el capítulo anterior se explicó el funcionamiento de la metodología *Scrum*. Este capítulo muestra la implementación de dicho método en la realización de la aplicación.

4.1 Product Backlog

Durante el *Product Backlog* se planificó la estructura general de la aplicación, se estableció una normativa para que el código fuente fuese consistente, se planteó un prototipo de interfaz para cada uno de los módulos que interactúan directamente con el usuario y se definieron los *Sprints* que se realizaron. A continuación se muestran los resultados de dicha planificación:

4.1.1 Estructura de la aplicación

Cada uno de los cuatro módulos de la aplicación está ubicado en un espacio de nombre (namespace) diferente y nunca interactúan entre sí. Adicionalmente se creó un namespace llamado *ClassLibrary* para colocar las clases útiles para todos los módulos (ver Figura 4.1).

Los cuatro módulos de la aplicación son:

- Servidor (wradvs).
- Herramienta de configuración del servidor (Server Configuration Tool).
- Visor de eventos (Log Viewer).
- Herramienta de configuración de direcciones IPv6 (IPv6 Address Configuration Tool).

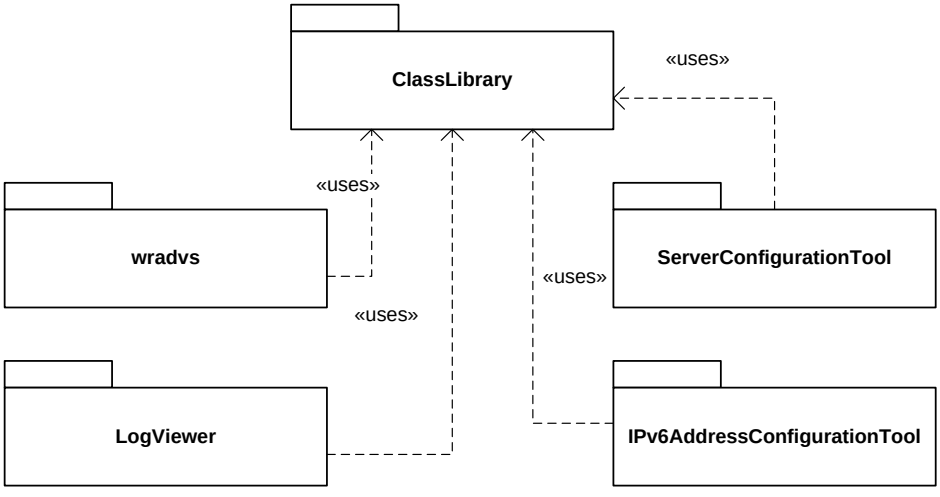

Figura 4.1: Diagrama de espacios de nombre.

En la Figura 4.2 se muestra el diagrama de casos de uso que refleja la interacción del usuario con el sistema.

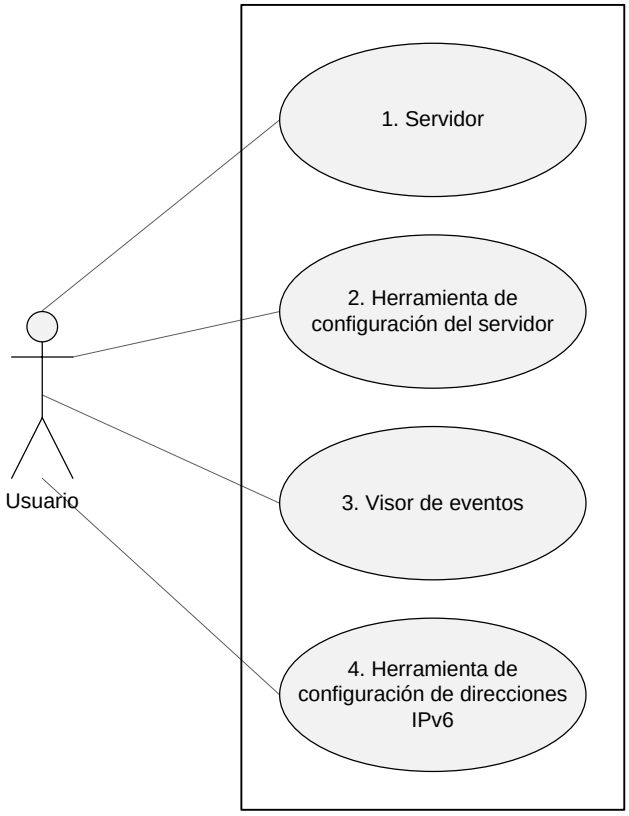

Figura 4.2: Diagrama de casos de uso - nivel 0.

Se muestra la especificación de cada caso de uso en las tablas: Tabla 4.2, Tabla 4.3, Tabla 4.4.

Caso de uso
1. Servidor
Actores
✓ Usuario.
Descripción
✓ Provee un servicio de Windows que envía mensajes RA periódicos y en respuesta a mensajes RS.
Puntos de exclusión
✓ Iniciar servidor. ✓ Detener servido. ✓ Reiniciar servidor.

Tabla 4.1: Especificación de caso de uso 1. Servidor

Caso de uso
2. Herramienta de configuración del servidor
Actores
✓ Usuario.
Descripción
✓ Proporciona una interfaz gráfica para configurar todos los parámetros y opciones de los mensajes RA.
Puntos de exclusión
✓ Seleccionar interfaz de red. ✓ Editar parámetros de interfaz. ✓ Configurar anuncio de prefijos. ✓ Configurar anuncio de servidores DNS recursivos. ✓ Configurar anuncio de rutas más específicas. ✓ Editar parámetros de Mobile IPv6. ✓ Iniciar, detener, reiniciar servidor. ✓ Cambiar idioma.

Tabla 4.2: Especificación de caso de uso 2 de la Herramienta de configuración del servidor.

Caso de uso
3. Visor de eventos
Actores
✓ Usuario.
Descripción
✓ Provee una interfaz gráfica para acceso en tiempo real al registro de eventos del servidor.
Puntos de exclusión
✓ Mostrar listado de eventos. ✓ Seleccionar evento. ✓ Seleccionar todos los eventos. ✓ Copiar información de eventos. ✓ Exportar listado de eventos. ✓ Borrar listado de eventos. ✓ Auto scroll. ✓ Cambiar idioma.

Tabla 4.3: Especificación de caso de uso 3 del Visor de eventos.

Caso de uso
4. Herramienta de configuración de direcciones IPv6
Actores
✓ Usuario.

Descripción
✓ Proporciona a los usuarios una interfaz gráfica para realizar las funciones de netsh relacionadas a la configuración de direcciones IPv6.
Puntos de exclusión
✓ Seleccionar interfaz de red. ✓ Administrar tabla de enrutamiento. ✓ Administrar reenvío de paquetes. ✓ Cambiar idioma.

Tabla 4.4: Especificación de caso de uso 4 de la Herramienta de configuración de direcciones IPv6.

El diagrama de clases general de la aplicación se puede apreciar en la Figura 4.3. Allí, se pueden observar las cinco librerías principales de la aplicación mencionadas al comienzo de este capítulo. La clase *Util* y *MyToolTips* son utilizadas por las clases principales (clase *Server* y clase *MainWindow*).

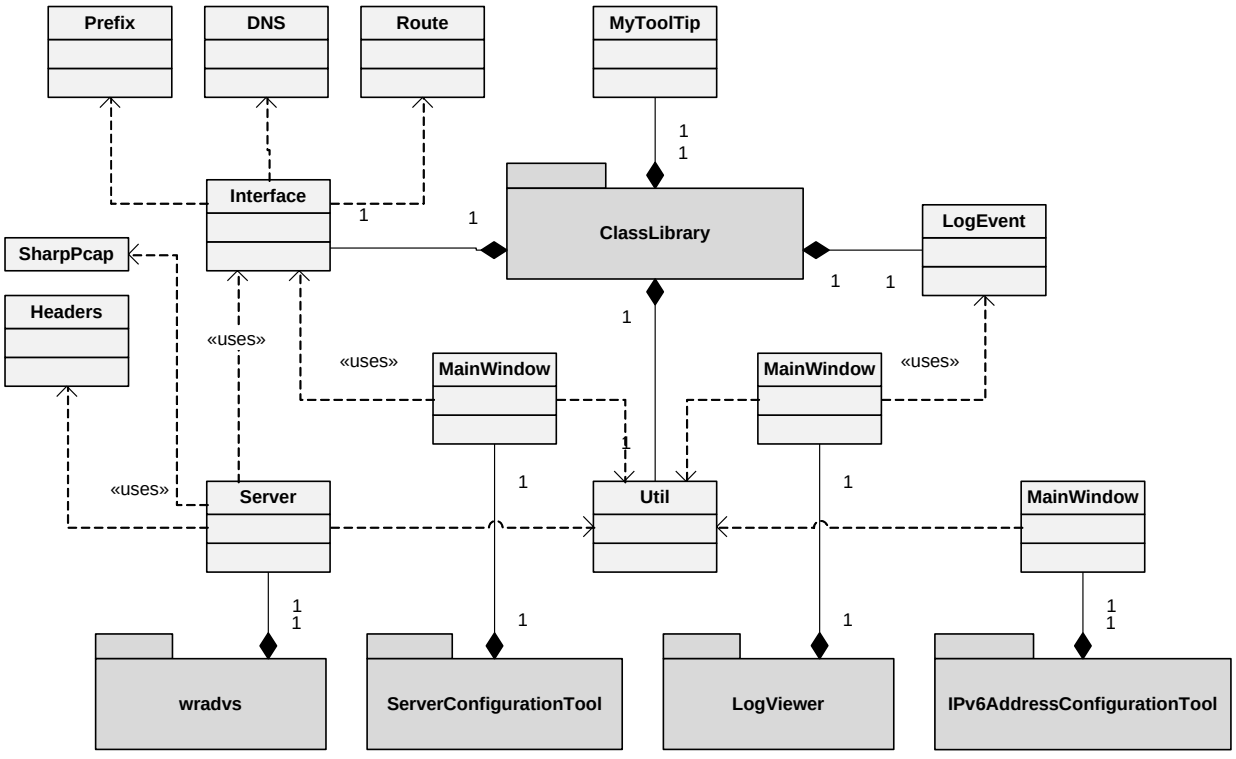

Figura 4.3: Diagrama general de clases.

4.1.2 Normativas generales

En este punto se establecieron normas para que el código fuente mantuviera el mismo estilo en toda la aplicación independientemente del programador. Las normas son las siguientes:

- ✓ Todas las clases, variables, métodos y comentarios se harán en idioma inglés.

- ✓ La primera letra de cada palabra que compone el nombre de una clase debe ser colocada en mayúscula (ver Figura 4.4 para un ejemplo).

```
public class Header
public class ICMPv6Header : Header
```

Figura 4.4: Ejemplo de normativa.

- ✓ Los métodos (funciones y acciones), variables y propiedades tendrán la inicial de la primera palabra en minúscula y el resto de las iniciales de cada palabra en mayúscula (ver Figura 4.5 para un ejemplo).

```
private bool fillAddressList()
public bool advOnLinkFlag;
```

Figura 4.5: Ejemplo de normativa.

- ✓ Los comentarios se harán en la línea superior a la instrucción o instrucciones que se desea aclarar (ver Figura 4.6 para un ejemplo).

```
//renews the nic's array
nics = NetworkInterface.GetAllNetworkInterfaces();
```

Figura 4.6: Ejemplo de normativa.

- ✓ La llave inicial de un bloque de código se colocará en la línea siguiente a la instrucción y se respetará el tabulado de cuatro (4) espacios en las instrucciones del cuerpo (ver Figura 4.7 para un ejemplo).

```
if (idfound)
{
    address += word;
}
```

Figura 4.7: Ejemplo de normativa.

4.1.3 Prototipo de interfaz

Durante el *Product Backlog* se establecieron lineamientos generales para construir las interfaces gráficas del sistema. Entre estos se encuentran:

- ✓ Todos los botones deben tener un tamaño de 65 px de ancho por 22 px de alto.

- ✓ Todas las ventanas deben poseer una barra de menú (*MenuBar*) en la parte superior.

- ✓ Se realizaron tres interfaces gráficas principales (herramienta de configuración del servidor, visor de eventos, herramienta de configuración de direcciones IPv6) y una ventana para los detalles del visor de eventos. El tamaño de las ventanas puede variar de acuerdo a las necesidades que se presentan, pero respetando los patrones mencionados anteriormente.

- ✓ Cualquier ventana adicional será de tipo *Tool Window* (no posee botón para minimizar y maximizar). Deben poseer dos botones en la parte inferior derecha (para aceptar o cancelar).

- ✓ En la Figura 4.8 se muestra el prototipo de interfaz para la herramienta de configuración del servidor.

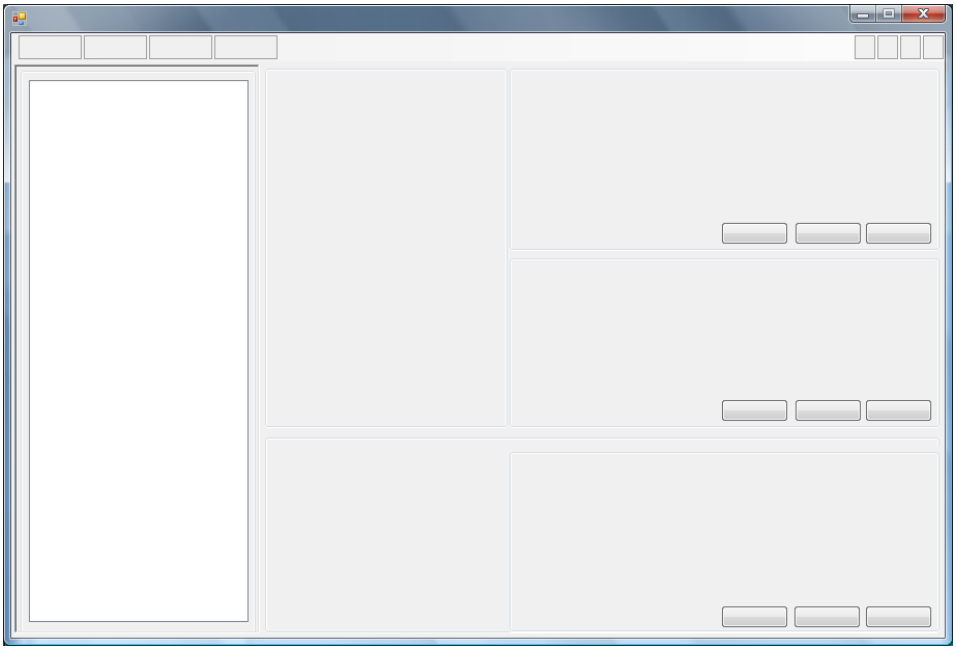

Figura 4.8: Prototipo de interfaz de la herramienta de configuración del servidor.

- ✓ En la Figura 4.9 se muestra el prototipo de interfaz para el visor de eventos.

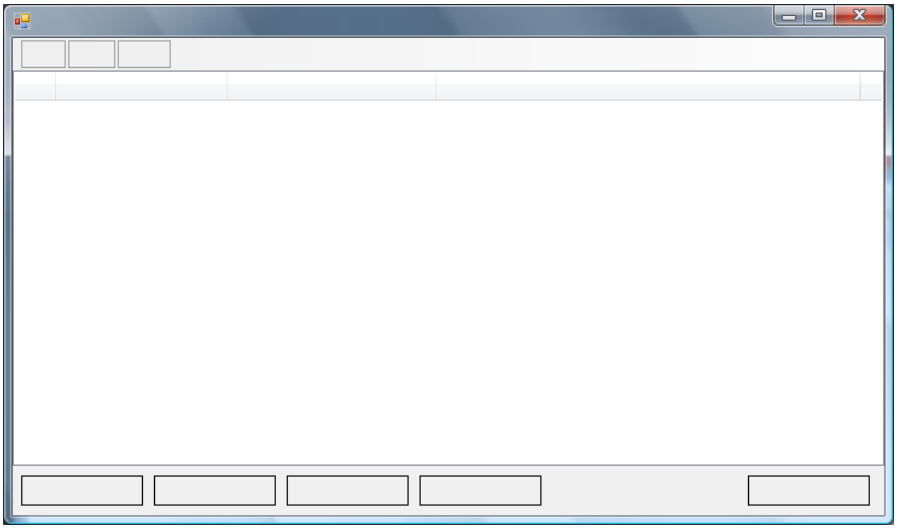

Figura 4.9: Prototipo de interfaz del visor de eventos.

✓ En la Figura 4.10 se muestra el prototipo de interfaz de la ventana de detalles del visor de eventos.

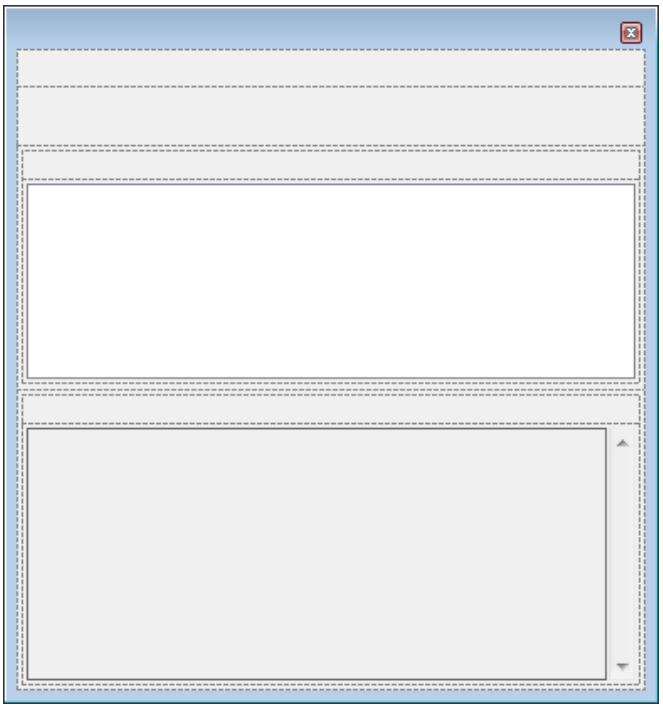

Figura 4.10: Prototipo de interfaz de la ventana de detalles del visor de eventos.

✓ En la Figura 4.11 se muestra el prototipo de interfaz para la herramienta de configuración de direcciones IPv6.

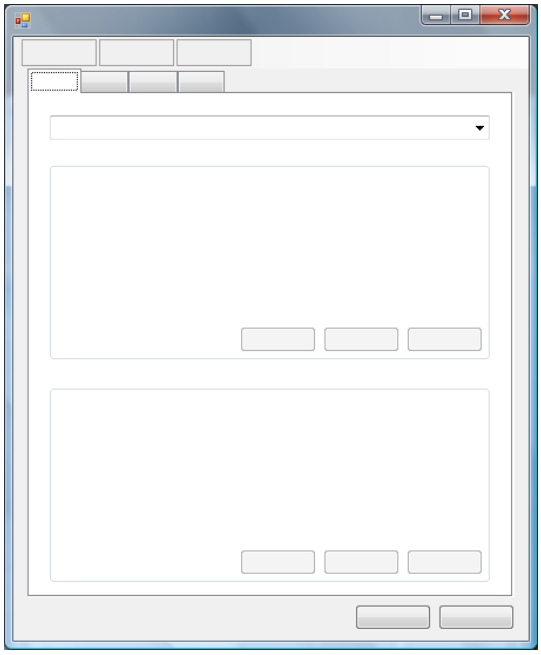

Figura 4.11: Prototipo de interfaz de la herramienta de configuración de direcciones IPv6.

4.1.4 Definición de Sprints

Durante el *Product Backlog* se establecieron cuales debían ser los *Sprints* necesarios para llevar a cabo la aplicación. Si durante la realización de un *Sprint* surgen nuevos requerimientos, estos se realizan durante el *Sprint Retrospective*.

	Duración	Descripción
Sprint 1	4 semanas	Desarrollo del servidor. Ir al capítulo 5.
Sprint 2	3 semanas	Desarrollo de la herramienta de configuración del servidor. Ir al capítulo 6.
Sprint 3	3 semanas	Desarrollo del visor de eventos. Ir al capítulo 7.
Sprint 4	4 semanas	Desarrollo de la herramienta para la configuración de direcciones IPv6. Ir al capítulo 8.
Sprint 5	1 semana	Incorporación de Tooltips (mensajes de ayuda al usuario). Ir al capítulo 9.
Sprint 6	2 semanas	Pruebas. Ir al capítulo 10.

Tabla 4.5: Definición de Sprints.

Los *Sprints* propuestos serán explicados a continuación en diferentes capítulos según se menciona en la descripción de la Tabla 4.5.

5. Servidor

Este capítulo describe el *Sprint 1* que explica la creación del servidor (wradvs), un servicio de Windows que se encarga de envíar mensajes RA periódicos y en respuesta a mensajes RS.

5.1 Sprint Planning Meeting

Se estipuló un tiempo de cuatro semanas para realizar este módulo. En este tiempo se construyó la estructura de los PDUs (Protocol Data Unit) que utilizó el servidor para enviar los mensajes RA de acuerdo a los RFCs correspondientes. El servidor es instalado como un servicio de Windows de modo que ofrece las opciones de iniciar, detener y reiniciar desde la ventana de administración de servicios de Windows.

La interacción que puede tener el usuario con el servidor a través de la ventana de administración de servicios de Windows se puede observar en el diagrama de casos de uso de la Figura 5.1.

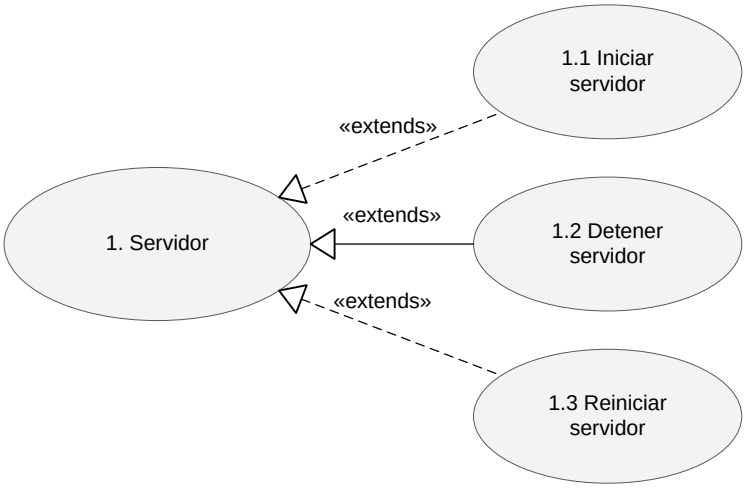

Figura 5.1: Diagrama de casos de uso 1 - nivel 1 del Servidor.

Se muestra la especificación de cada caso de uso para el servidor en las tablas: Tabla 5.1, Tabla 5.2, Tabla 5.3.

Caso de uso
1.1 Iniciar servidor
Actores
✓ Usuario.

Descripción
Se inicia la ejecución del servidor.
Flujo básico
✓ El usuario accede a la opción “Herramientas” -> “Iniciar Servidor” de la herramienta de configuración del servidor. ✓ El usuario inicia el servicio wradvs (el servidor) por medio de la herramienta *services.msc* de Windows.

Tabla 5.1: Especificación de caso de uso 1.1 Iniciar servidor.

Caso de uso
1.2 Detener servidor
Actores
✓ Usuario.
Descripción
Se detiene la ejecución del servidor.
Flujo básico
✓ El usuario accede a la opción “Herramientas” -> “Detener Servidor” de la herramienta de configuración del servidor. ✓ El usuario detiene el servicio wradvs (el servidor) por medio de la herramienta *services*.msc de Windows.

Tabla 5.2: Especificación de caso de uso 1.2 Detener servidor.

Caso de uso
1.3 Reiniciar servidor
Actores
✓ Usuario.
Descripción
Se reinicia la ejecución del servidor.
Flujo básico
✓ El usuario accede a la opción “Herramientas” -> “Reiniciar Servidor” de la herramienta de configuración del servidor. ✓ El usuario reinicia el servicio wradvs (el servidor) por medio de la herramienta *services.msc* de Windows.

Tabla 5.3: Especificación de caso de uso 1.3 Reiniciar servidor.

El primer paso fue realizar la estructura de los PDUs, tras una investigación se concluyó que serían necesarios los siguientes PDUs:

- Internet Protocol version 6 [2].
- Router Advertisement (ICMPv6) [12].
- Prefix Information Option [12].
- Source Link Layer Address Option [12].

- MTU Option [12].
- Route Information Option [13].
- Recursive DNS Server Option [15].
- Advertisement Interval Option [14].
- Home Agent Information Option [16].

Se utilizó la clase *ServiceBase* y *ServiceInstaller* de C# para configurar el servicio de Windows, dado que el lenguaje proporciona los métodos y propiedades para facilitar el proceso.

Para realizar el envío de mensajes RA, se utilizó la clase *Sockets* de C# y para recibir cualquier mensaje por las interfaces de red se utilizó un *wrapper* para C# de la librería *WinPcap* llamado *SharpPcap*[5] que proporciona los mecanismos para mantenerse a la escucha de paquetes en las diferentes interfaces.

Se creó un log de eventos de Windows utilizando la clase *EventLog* de C# en el que se almacenaron todos los mensajes de información y error que serían utilizados más adelante por el visor de eventos (ver capítulo 7).

Una vez que se investigaron todos los métodos relacionados a las librerías previamente mencionadas para la realización de este módulo se inició el *Sprint Backlog*.

5.2 Sprint Backlog

El primer paso que se realizó durante el *Sprint Backlog*, como se mencionó anteriormente, fue crear la estructura de los PDUs que se muestra en el diagrama de clases en la Figura 5.2. Posteriormente se realizó el servidor (ver diagrama en la Figura 5.3) que se encarga de generar los mensajes utilizando las estructuras creadas.

[5] http://sourceforge.net/projects/sharppcap

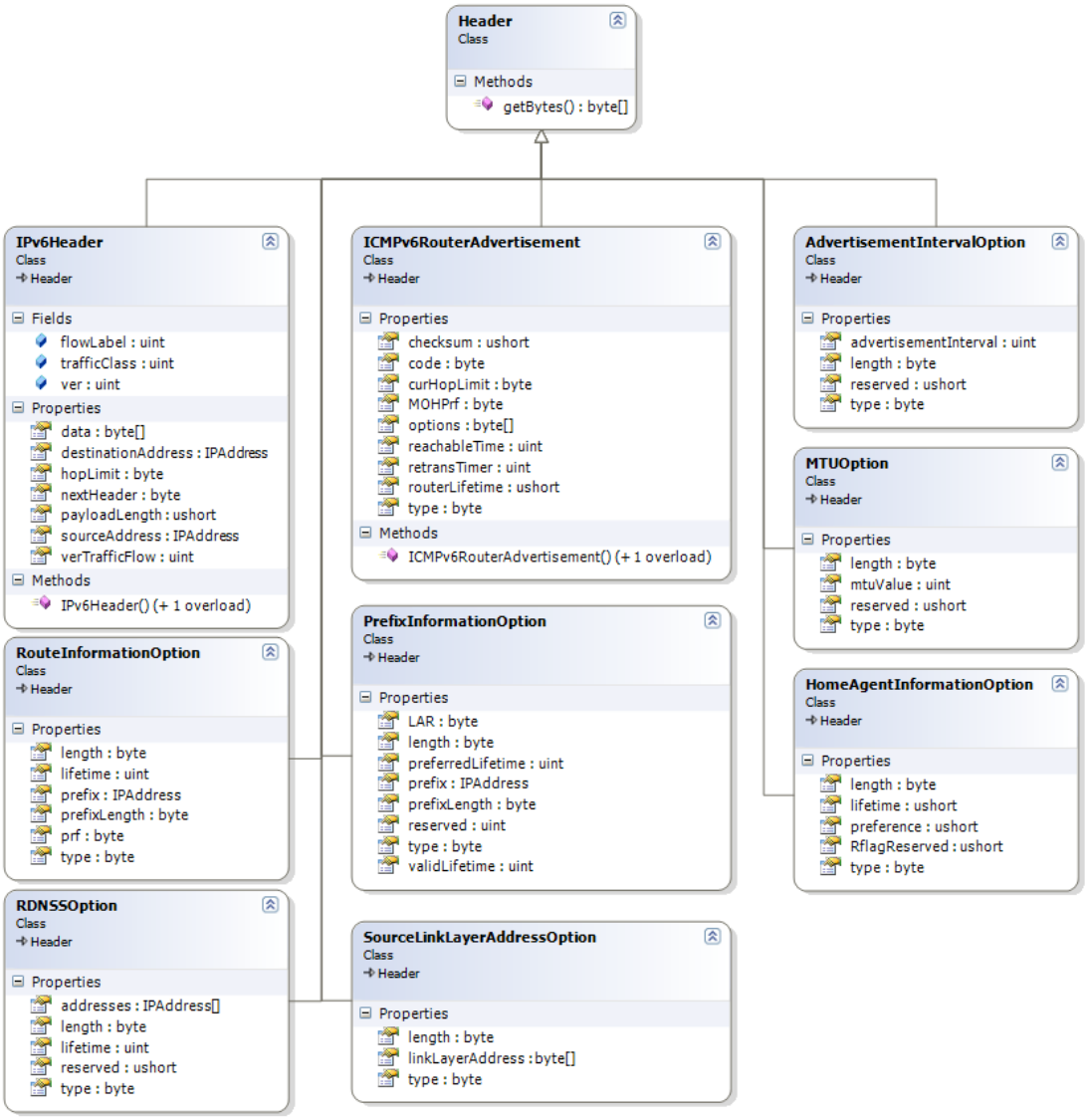

Figura 5.2: Diagrama de clases para los PDU.

La clase *Header* (ver Figura 5.2) posee el método *getBytes*, cuya función es transformar el objeto que la invoca en un arreglo de bytes y así poder enviar el mensaje a través de un socket.

Todas las clases del diagrama menos la clase *Header*, son equivalentes a un mensaje definido en algún RFC y las propiedades existentes dentro de las clases son los campos del mensaje.

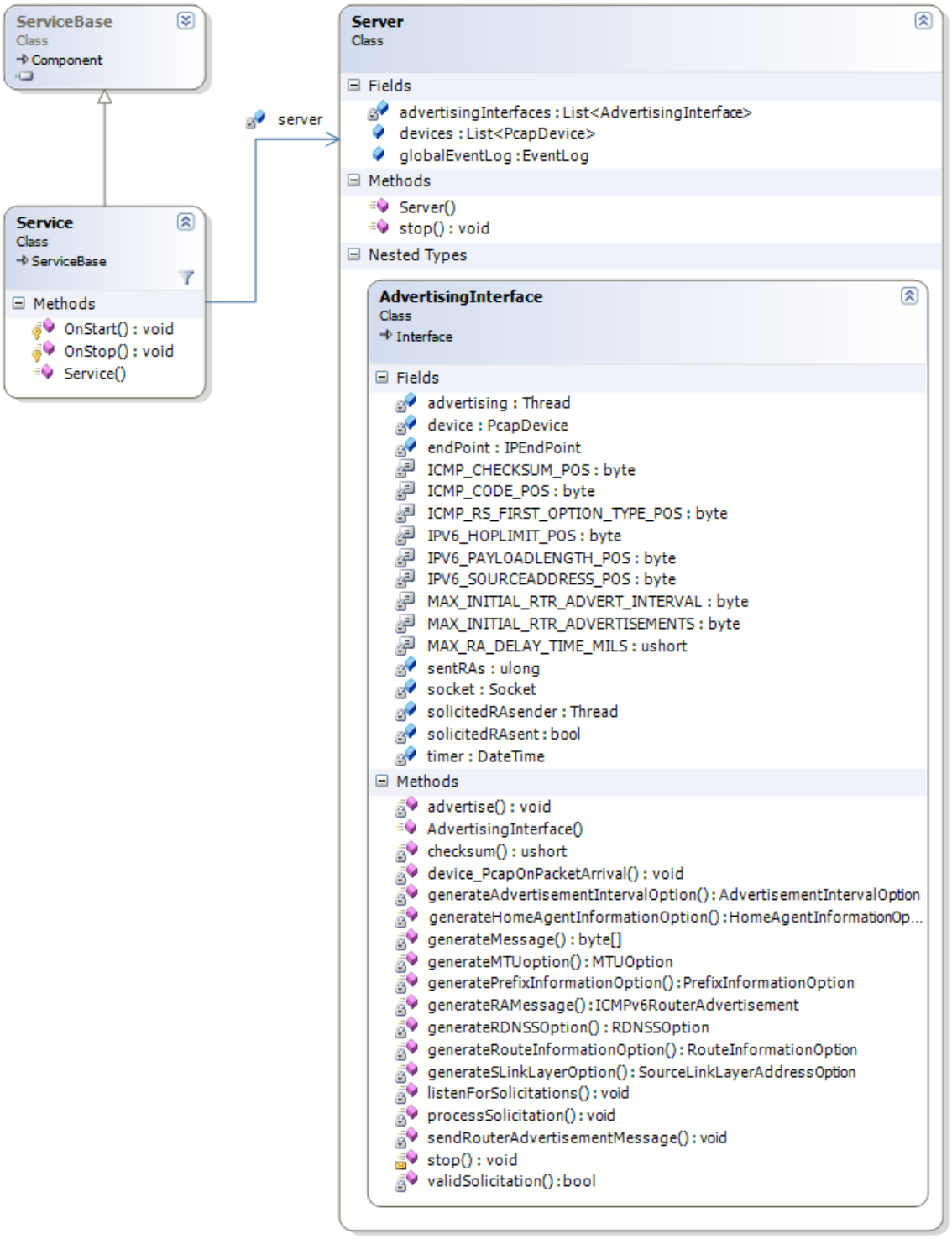

Figura 5.3: Diagrama de clases para el servidor (wradvs).

Entre las propiedades más destacadas que se pueden observar en la Figura 5.2 se encuentran:

- ✓ La propiedad *data* del *IPv6Header* que consiste en un arreglo de bytes obtenido con la función *getBytes* de un objeto de tipo *ICMPv6RouterAdvertisement*.

- ✓ La propiedad *verTrafficFlow* (ver Figura 5.4) de la clase *IPv6Header* es un conjunto de tres campos (*Version*, *Traffic Class*, *Flow Label*) y no un campo individual como el resto de las propiedades.

```
1  public UInt32 verTrafficFlow
2  {
3      get
4      {
5          return (UInt32)((ver << 28) | (trafficClass << 20) | flowLabel);
6      }
7  }
```

Figura 5.4: Composición de la propiedad verTrafficFlow.

Una vez definidas las estructuras, estas son utilizadas desde el servidor (clase *Server*) que posee los métodos necesarios para generar el mensaje RA (ver Figura 5.3) tomando en cuenta los parámetros de configuración existentes en el archivo *configuration.xml* (ver Figura 5.5). Este archivo puede ser modificado utilizando la herramienta de configuración del servidor (ver capítulo 6).

El servidor posee dos hilos en ejecución, uno que calcula el tiempo que se debe esperar entre dos mensajes RA (mensajes RA periódicos) y un hilo que se mantiene a la escucha de mensajes RS. Cuando un mensaje RS es recibido, el hilo que controla los RA es interrumpido y se genera el nuevo mensaje RA en respuesta al mensaje RS. Una vez enviado el mensaje RA se recalcula el tiempo de transmisión de los RA periódicos.

Para el envío de los mensajes se utilizó la clase *Socket*, específicamente la función *sendTo* que toma un arreglo de bytes y la dirección destino que en este caso es la dirección *multicast* de todos los nodos (ff02::1).

```
1  <?xml version="1.0"?>
2  <Interfaces>
3    <Interface id="{50733E78-5FE7-46F0-8E51-02BAB9CEC4AA}"
4     name="Wireless Network Connection" index="8">
5      <AdvSendAdvertisements>False</AdvSendAdvertisements>
6      <AdvSourceLLAddress>True</AdvSourceLLAddress>
7      <MaxRtrAdvInterval>600</MaxRtrAdvInterval>
8      <MinRtrAdvInterval>198</MinRtrAdvInterval>
9      <AdvManagedFlag>False</AdvManagedFlag>
```

```
10      <AdvOtherConfigFlag>False</AdvOtherConfigFlag>
11      <AdvHomeAgentFlag>False</AdvHomeAgentFlag>
12      <AdvDefaultPreference>1</AdvDefaultPreference>
13      <AdvLinkMtu>0</AdvLinkMtu>
14      <AdvReachableTime>0</AdvReachableTime>
15      <AdvRetransTimer>0</AdvRetransTimer>
16      <AdvCurHopLimit>64</AdvCurHopLimit>
17      <AdvDefaultLifetime>1800</AdvDefaultLifetime>
18      <AdvHomeAgentInfo>False</AdvHomeAgentInfo>
19      <HomeAgentLifetime>1800</HomeAgentLifetime>
20      <HomeAgentPreference>0</HomeAgentPreference>
21      <AdvMobRtrSupportFlag>False</AdvMobRtrSupportFlag>
22      <AdvIntervalOpt>False</AdvIntervalOpt>
23      <AdvPrefixList />
24    </Interface>
25  </Interfaces>
```

Figura 5.5: Ejemplo del archivo configuration.xml.

A medida que se van enviando y recibiendo mensajes el servidor llena un log del sistema añadiendo entradas en un objeto de tipo *LogEvent*. Estas entradas pueden contener datos de los RA enviados, información sobre las interfaces que reciben mensajes RS y mensajes de error o información sobre el estado de las interfaces de red. Toda esta información almacenada es posteriormente utilizada por el visor de eventos (ver capítulo 7) para construir los mensajes que se muestran al usuario en tiempo real.

5.3 Sprint Review

Durante la revisión se analizó el código de los métodos más relevantes del módulo para conseguir fallas o mejorar procesos. Los métodos más relevantes fueron:

- Hilo para el envío de RA periódicos (ver Figura 5.6).
- Hilo a la escucha de mensajes RS (ver Figura 5.7).
- Hilo para procesar los mensajes RS e interrumpir el envío de mensajes RA periódicos (ver Figura 5.8).

```
1   private void advertise()
2   {
3       Random r = new Random(DateTime.Now.Millisecond);
4       int rnd;
5
6       globalEventLog.WriteEntry("{[" + index + "]" + name + "}event1");
7       while(true)
8       {
9           nic = Util.updateNetworkInterface(id);
10          if (nic == null)
11          {
12              globalEventLog.WriteEntry("{[" + index + "]" + name +"}error1");
13              stop();
```

```
14              return;
15          }
16          else
17          {
18              index = nic.GetIPProperties().GetIPv6Properties().Index;
19              name = nic.Name;
20          }
21          sendRouterAdvertisementMessage(Util.LogEvent.MsgType.RA);
22          do
23          {
24              lock (endPoint)
25              {
26                  //random value MinRtrAdvInterval and MaxRtrAdvInterval.
27                  rnd = r.Next((int)minRtrAdvInterval, (int)maxRtrAdvInterval);
28                  rnd /= (advHomeAgentFlag || advIntervalOpt) ? 1000 : 1;
29                /*For the first few advertisements (up to
30                 MAX_INITIAL_RTR_ADVERTISEMENTS) sent from an interface when
31                 it becomes an advertising interface, if the randomly chosen
32                 interval is greater than MAX_INITIAL_RTR_ADVERT_INTERVAL, the
33                 timer SHOULD be set to MAX_INITIAL_RTR_ADVERT_INTERVAL instead*/
34                  if (sentRAs < MAX_INITIAL_RTR_ADVERTISEMENTS &&
35                  rnd > MAX_INITIAL_RTR_ADVERT_INTERVAL)
36                      rnd = MAX_INITIAL_RTR_ADVERT_INTERVAL;
37                  solicitedRAsent = false;
38                  //setting the timer.
39                  timer = DateTime.Now.AddSeconds(rnd);
40              }
41              try
42              {
43                  Thread.Sleep(rnd * 1000);
44              }
45              catch (ThreadInterruptedException)
46              {
47                sendRouterAdvertisementMessage(Util.LogEvent.MsgType.SolicitedRA);
48                solicitedRAsent = true;
49              }
50          }
51          //in case a RS have been responded (with an RA)
52          //the timer should be recalculated.
53          while (solicitedRAsent);
54      }
55      //the interface keeps sending unsolicited RA messages
56  }
```

Figura 5.6: Código del método advertise (envío de mensajes RA).

En la línea 6 (Figura 5.6) se observa cómo es colocada una entrada en el objeto *globalEventLog* de tipo *EventLog*. La entrada registra el identificador, el nombre y el índice de la interfaz de red además de un código (en este caso *event1*), que indica al visor de eventos que mensaje de información o error debe ser mostrado en la lista.

Posteriormente comienza un ciclo infinito (línea 7, Figura 5.6), donde se utilizó el método *sendRouterAdvetisementMessage* (línea 21, Figura 5.6) para enviar el mensaje RA sin importar si es mensaje RA periódico o no.

En la línea 27 (Figura 5.6) se muestra como es calculado el tiempo que se debe esperar antes de enviar el siguiente mensaje RA. Este es un número aleatorio entre el tiempo de intervalo mínimo y máximo que el usuario haya configurado para el envío de mensajes RA periódicos.

Por último la instrucción *Thread.Sleep(rnd * 1000)* (línea 43, Figura 5.6) mantiene al servicio a la espera del tiempo calculado previamente, al culminar este tiempo se enviará un nuevo mensaje RA, a menos que sea interrumpido desde el hilo que recibe los mensajes RS. La Figura 5.7 muestra un fragmento de código del método *device_PcapOnPacketArrival*.

```
1  private void device_PcapOnPacketArrival(object sender, PcapCaptureEventArgs e)
2  {
3      IPv6Packet IPv6 = ((IPPacket)e.Packet).ipv6;
4      //verifies if IPv6 Next Header field indicates an ICMPv6 header is next.
5      if (IPv6.NextHeader == IPProtocol.IPProtocolType.ICMPV6)
6      {
7          //verifies if the ICMPv6 message is a Router Solicitation.
8          if (IPv6.IPData[0] == 133)
9          {
10             if (!IPv6.SourceHwAddress.Equals(e.Device.Interface.MacAddress))
11             {
12                 //write event in the log.
13                 string msg = HexHelper.toString(IPv6.IPv6Header) + " " +
14                              HexHelper.toString(IPv6.IPData);
15                 globalEventLog.WriteEntry("{[" + index + "]" + name +
16                                           "}RS " + msg);
17                 if (solicitedRAsender == null || !solicitedRAsender.IsAlive)
18                 {
19                     solicitedRAsender = new Thread(
20                         new ParameterizedThreadStart(processSolicitation));
21                     solicitedRAsender.Start(IPv6);
22                 }
23             }
24         }
25     }
26 }
```

Figura 5.7: Fragmento de código del método device_PcapOnPacketArrival.

El método *device_PcapOnPacketArrival* es un evento ejecutado cada vez que un paquete cualquiera llega por alguna interfaz de red. El método verifica cada mensaje para obtener aquellos que sean mensajes RS. La verificación consiste en tomar los paquetes IPv6 (línea 3, Figura 5.7), observar si el campo *NextHeader* indica que el siguiente encabezado es ICMPv6 (línea 5, Figura 5.7) y ver si el campo *Type* de la cabecera ICMPv6 es 133 (*Router Solicitation*. Línea 8, Figura 5.7). Al conseguir un mensaje RS se crea un hilo para su procesamiento (línea 20, Figura 5.7) invocando al método *processSolicitation* (Figura 5.8).

```
1  private void processSolicitation(object bytes)
2  {
3      //if is a valid RS.
4      if (validSolicitation(((IPv6Packet)bytes).IPv6Header,
5      ((IPv6Packet)bytes).IPData))
6      {
7          /*(1)Upon receipt of a Router Solicitation, compute a random delay
8              within the range 0 through MAX_RA_DELAY_TIME.
9            (2)If the computed value corresponds to a time later than the time
10             the next multicast Router Advertisement is scheduled to be sent
11           (3)ignore the random delay and send the advertisement at the
12             already-scheduled time.
13         */
14         /*(1)*/
15         int rnd = new Random(DateTime.Now.Millisecond).Next(
16                                 MAX_RA_DELAY_TIME_MILS);
17         /*(2)*/
18         if (DateTime.Now.AddMilliseconds(rnd).CompareTo(timer) < 0)
19         {
20             try
21             {
22                 if (advertising.ThreadState ==
23                 System.Threading.ThreadState.WaitSleepJoin)
24                     advertising.Interrupt();
25             }
26             catch
27             {
28                 //do nothing
29             }
30             //delaying
31             Thread.Sleep(rnd);
32         }
33     }
34 }
```

Figura 5.8: Código del método processSolicitation.

El método *advertise* es invocado desde un objeto global llamado *advertising*. El código entre las líneas 22 y 24 (Figura 5.8) verifica que el hilo que maneja el envío de mensajes RA se encuentre en estado *Sleep* y de ser así interrumpe ese estado para que un nuevo mensaje RA se envíe como respuesta al mensaje RS que ha llegado.

5.4 Sprint Retrospective

Durante el *Sprint Retrospective* se llevaron a cabo las siguientes actividades:

- Se añadió el soporte a Mobile IPv6.
- Se modificó la forma de acceso al idioma para hacerlo global en la aplicación.
- Se optimizó el proceso de envío de mensajes RA en respuesta a los mensajes RS

6. Herramienta de Configuración del Servidor

Este capítulo describe el *Sprint 2*. La herramienta de configuración del servidor (wradvs) proporciona una interfaz gráfica para configurar todos los parámetros y opciones de los mensajes RA. Permite iniciar, detener y reiniciar el servicio de envío de mensajes RA sin necesidad de acceder a la ventana de servicios de Windows.

6.1 Sprint Planning Meeting

Durante la planificación se especificaron todos los servicios que debe proveer la herramienta de configuración del servidor, estos se muestran en el diagrama de casos de la ver Figura 6.1.

La herramienta de configuración del servidor se encarga de modificar el archivo *configuration.xml* utilizado por el servidor para la creación de los mensajes RA. Para hacer las modificaciones se creó una interfaz gráfica que posee todos los parámetros y opciones en una sola ventana para que el usuario acceda a los campos con facilidad.

Se contemplaron todos los posibles errores de configuración, es decir, que el usuario no pueda colocar valores inválidos en los campos de texto. Se validó tanto los limites de cada valor como que dos campos relacionados no posean valores incongruentes.

Para indicar al usuario los posibles valores de un campo se muestra un tooltip cuando se arrastra el ratón sobre el. Para evitar que los usuarios cometan errores, los caracteres son validados a medida que son agregados y de ser inválidos no se agregan, mostrando un tooltip que indica los caracteres aceptados por el campo. Si un valor excede los límites o no respeta los formatos el campo es colocado en color rojo. Si se trata de iniciar el servicio con errores en el archivo de configuración el servidor se encarga de mostrar en qué campos existen errores para que el usuario se encargue de verificarlos.

El diagrama de casos de uso de la Figura 6.1 muestra el conjunto de acciones que se pueden llevar a cabo en la herramienta de configuración del servidor.

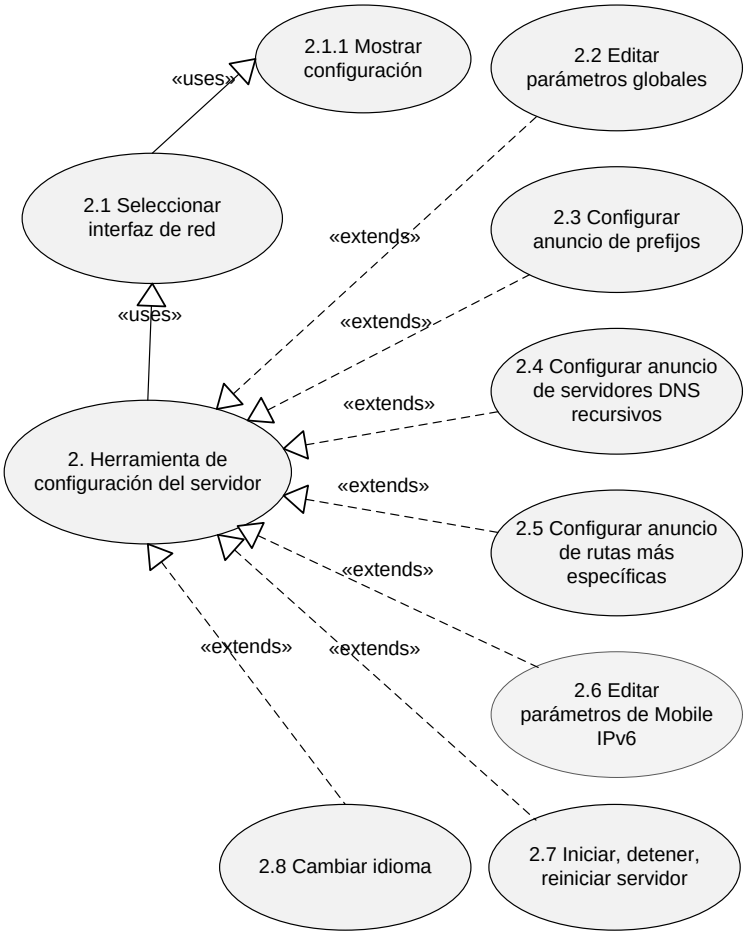

Figura 6.1: Diagrama de casos de uso 2 - nivel 1. Herramienta de configuración del servidor.

Las especificaciones de los casos de uso se observan en las tablas: Tabla 6.1, Tabla 6.2, Tabla 6.3, Tabla 6.4, Tabla 6.5, Tabla 6.6, Tabla 6.7, Tabla 6.8, Tabla 6.9

Caso de uso
2.1 Seleccionar interfaz de red
Actores
✓ Usuario.
Descripción
✓ Se presenta una lista con las interfaces activas con IPv6.
Flujo básico
✓ Seleccionar una interfaz de la lista.
Puntos de inclusión
✓ Mostrar configuración.

Tabla 6.1: Especificación de caso de uso 2.1 Seleccionar interfaz de red.

Caso de uso
2.1.1 Mostrar configuración
Actores
✓ Usuario.
Descripción
✓ Se muestra la configuración se la interfaz seleccionada.
Flujo básico
✓ Se cargan las opciones y parámetros especificados en dicha interfaz. ✓ Si no se ha hecho ninguna configuración se colocan los parámetros por defecto especificado por los RFCs.
Puntos de inclusión
✓ Mostrar configuración.

Tabla 6.2: Especificación de caso de uso 2.1.1 Mostrar configuración.

Caso de uso
2.2 Editar parámetros de interfaz
Actores
✓ Usuario.
Descripción
✓ Se muestran diferentes casillas de activación (*checkbox*), etiquetas (label) y cajas de texto (*textbox*) para configurar los parámetros de un mensaje RA.
Flujo básico
✓ Activar/desactivar envío de anuncios. ✓ Activar/desactivar envío de dirección de capa de enlace. ✓ Introducir el intervalo máximo de transmisión de mensajes RA. ✓ Introducir el intervalo mínimo de transmisión de mensajes RA. ✓ Activar/desactivar la bandera de configuración vía DHCPv6. ✓ Activar/desactivar la bandera de configuración de otros parámetros. ✓ Introducir el MTU del enlace. ✓ Introducir el tiempo alcanzable. ✓ Introducir el tiempo de retransmisión. ✓ Indicar el límite de saltos.

Tabla 6.3: Especificación de caso de uso 2.2 Editar parámetros de interfaz.

Caso de uso
2.3 Configurar anuncio de prefijos
Actores
✓ Usuario.
Descripción
✓ Se muestra listado de prefijos y tres botones para añadir, editar y/o eliminar prefijos de la lista.

Flujo básico
✓ Si se presiona el botón “Agregar”, se muestra una ventana para introducir los datos necesarios para añadir un nuevo prefijo. ✓ Si se selecciona uno de los elementos de la lista y se presiona el botón “Editar” o se hace doble-clic, se muestra una ventana para modificar la información del prefijo seleccionado. ✓ Si se selecciona un elemento de la lista y se presiona el botón “Remover”, se elimina el prefijo seleccionado.

Tabla 6.4: Especificación de caso de uso 2.3 Configurar anuncio de prefijos.

Caso de uso
2.4 Configurar anuncio de servidores DNS recursivos.
Actores
✓ Usuario.
Descripción
✓ Se muestra listado de servidores y tres botones para añadir, editar y/o eliminar prefijos de la lista.
Flujo básico
✓ Si se presiona el botón “Agregar”, se muestra una ventana para introducir los datos necesarios para añadir un nuevo servidor DNS recursivo. ✓ Si se selecciona uno de los elementos de la lista y se presiona el botón “Editar” o se hace doble-clic, se muestra una ventana para modificar la información del servidor DNS recursivo seleccionado. ✓ Si se selecciona un elemento de la lista y se presiona el botón “Remover”, se elimina el servidor DNS recursivo seleccionado.

Tabla 6.5: Especificación de caso de uso 2.4 Configurar anuncio de servidores RDNS.

Caso de uso
2.5 Configurar anuncio de rutas más específicas
Actores
✓ Usuario.
Descripción
✓ Se muestra listado de rutas más específicas y tres botones para añadir, editar y/o eliminar prefijos de la lista.
Flujo básico
✓ Si se presiona el botón “Agregar”, se muestra una ventana para introducir los datos necesarios para añadir una nueva ruta. ✓ Si se selecciona uno de los elementos de la lista y se presiona el botón “Editar” o se hace doble-clic, se muestra una ventana para modificar la información de la ruta seleccionada. ✓ Si se selecciona un elemento de la lista y se presiona el botón “Remover”, se elimina la ruta seleccionada.

Tabla 6.6: Especificación de caso de uso 2.5 Configurar anuncio de rutas más específicas.

Caso de uso
2.6 Editar parámetros de Mobile IPv6
Actores
✓ Usuario.
Descripción
✓ Se muestran diferentes cajas de activación (*checkbox*), etiquetas (*labels*) y cajas de texto (*textbox*) para configurar los parámetros de Mobile IPv6.
Flujo básico
✓ Activar/desactivar la bandera de Home Agent. ✓ Activar/desactivar envío de la opción Home Agent Information. ✓ Introducir el tiempo de vida de Home Agent. ✓ Introducir la preferencia de Home Agent. ✓ Activar/desactivar soporte de movilidad de router. ✓ Activar/desactivar envío de la opción Advertisement Interval.

Tabla 6.7: Especificación de caso de uso 2.6 Editar parámetros de Mobile IPv6.

Caso de uso
2.7 Iniciar, detener, reiniciar servidor
Actores
✓ Usuario.
Descripción
Se ofrece al usuario un menú para iniciar, detener o reiniciar el servidor.
Flujo básico
✓ El usuario accede a la opción “Herramientas” -> “Iniciar Servidor”, para iniciar el servicio de Windows correspondiente al servidor. ✓ El usuario accede a la opción “Herramientas” -> “Detener Servidor”, para detener el servicio de Windows correspondiente al servidor. ✓ El usuario accede a la opción “Herramientas” -> “Reiniciar Servidor”, para reiniciar el servicio de Windows correspondiente al servidor.

Tabla 6.8: Especificación de caso de uso 2.8 Iniciar, detener, reinicias servidor.

Caso de uso
2.8 Cambiar idioma
Actores
✓ Usuario.
Descripción
Se ofrece al usuario un menú para modificar el lenguaje de la aplicación.
Flujo básico
✓ El usuario accede a la opción “Herramientas” -> “Idioma” y selecciona una de las dos opciones que se presentan (“English” o “Español”). ✓ Al hacer clic sobre la opción se reinicia la aplicación y utiliza ahora el idioma seleccionado.

Tabla 6.9: Especificación de caso de uso 2.7 Cambio de idioma.

Al culminar el análisis de los requerimientos de esta herramienta se inicializó el proceso de creación y diseño en el *Sprint Backlog*.

6.2 Sprint Backlog

En esta etapa del desarrollo se creó la herramienta de configuración del servidor. La herramienta consta de una ventana principal en la que al seleccionar una interfaz de red, se activa la configuración de los parámetros y opciones que se envían en los mensajes RA.

Esta herramienta genera un archivo XML de configuración llamado *configuration.xml* utilizado por el servidor para generar los mensajes RA. Adicionalmente, permite a los usuarios iniciar, detener o reiniciar el servicio de envío de mensajes.

Existen diferentes opciones y parámetros que pueden ser configurados con la herramienta. Entre los parámetros están: parámetros de interfaz, parámetros para especificaciones del router y parámetros para movilidad en IPv6. Entre las opciones se encuentran: anuncio de prefijos, anuncio de servidores DNS recursivos y anuncio de rutas más específicas.

Los parámetros de interfaz son:

- Enviar anuncios (AdvSendAdvertisements).
- Enviar dirección de capa de enlace (AdvSourceLLAddress).
- Intervalo máximo (MaxRtrAdvInterval).
- Intervalo mínimo (MinRtrAdvInterval).
- Configurar opciones con DHCPv6 (AdvManagedFlag).
- Otros parámetros de configuración (AdvOtherConfigFlag).
- MTU del enlace (AdvLinkMTU).
- Tiempo alcanzable (AdvReachableTime).
- Tiempo de retransmisión (AdvRetransTimer).
- Límite de saltos (AdvCurHopLimit).

Los parámetros para especificaciones del router son:

- Tiempo de vida (AdvDefaultLifetime).
- Preferencia del router (AdvDefaultPreference).

Los parámetros para movilidad en IPv6 son:

- Home agent (AdvHomeAgentFlag).
- Enviar opción Home agent information (AdvHomeAgentInfo).
- Tiempo de vida (HomeAgentLifetime).
- Preferencia (HomeAgentPreference).

- Soporte de movilidad de router (AdvMobRtrSupportFlag).
- Enviar la opción de intervalo de anuncios (AdvIntervalOpt).

La opción de anuncio de prefijos permite añadir una lista de prefijos (*AdvPrefixList*) para ser anunciados en el enlace, donde cada prefijo posee los siguientes parámetros:

- Prefijo (AdvPrefix).
- Longitud del prefijo (AdvPrefixLegth).
- Bandera dentro del enlace (AdvOnLinkFlag).
- Bandera de autoconfiguración sin estado (AdvAutonomousFlag).
- Bandera Router Address (AdvRouterAddrFlag).
- Tiempo de vida (AdvValidLifetime).
- Tiempo preferido (AdvPreferredLifetime).

La opción de anuncio de servidores DNS recursivos permite añadir una lista de servidores (*AdvRDNSSList*) con las siguientes especificaciones para cada entrada:

- Dirección del servidor (AdvDNSAddress).
- Tiempo de vida (AdvDNSLifetime).

La opción de anuncio de rutas más específicas permite colocar una lista de rutas (*AdvRouteList*) en la opción *Route Information* de los mensajes RA y cada ruta posee los siguientes parámetros:

- Prefijo (AdvRoutePrefix).
- Longitud del prefijo (AdvRoutePrefixLength).
- Tiempo de vida (AdvRouteLifetime).
- Preferencia (AdvRoutePreference).

Los parámetros y las opciones mencionados son enviados en los mensajes RA por alguna interfaz de red, por lo que se creó una clase llamada *Interface* con una variable por cada parámetro y una lista para cada opción.

Cuando se carga la ventana *MainWindow* una lista de objetos de tipo *Interface* se llena con los parámetros leídos en el archivo *configuration.xml*. Si no se posee ninguna configuración para la interfaz seleccionada, los parámetros toman los valores por defecto que indican los RFCs y no se colocan entradas para las opciones.

En el diagrama de clases de la Figura 6.2 se muestra la estructura de la clase *Interface*:

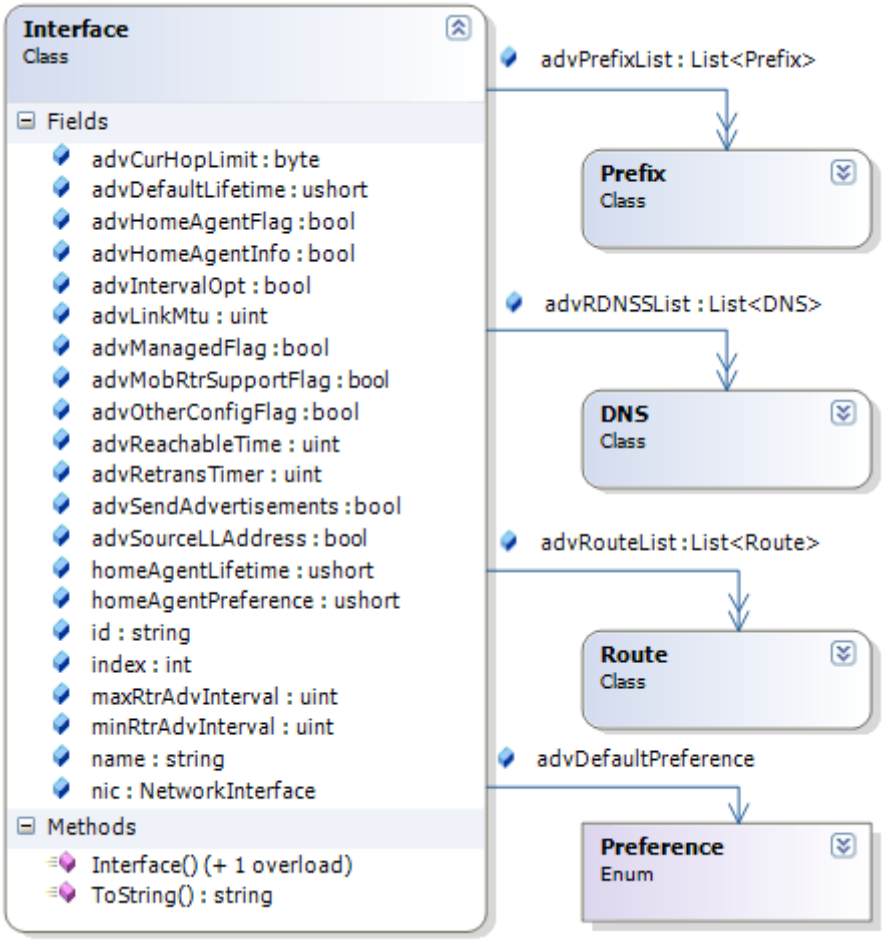

Figura 6.2: Diagrama de clases para *Interface*.

Como se puede observar en la Figura 6.2, la clase *Interface* tiene una variable llamada *nic* que posee la información de la interfaz de red. Con esa información se llenan las variables *id*, *index* y *name* que servirán para identificar a la interfaz. El resto de las variables se corresponde con un campo de los mensajes RA. Las clases *Prefix*, *DNS* y *Route* son explicadas en el diagrama de clases de la Figura 6.3 que contiene la estructura de la ventana principal.

En la Figura 6.3 se muestra el diagrama de clases de la herramienta de configuración del servidor.

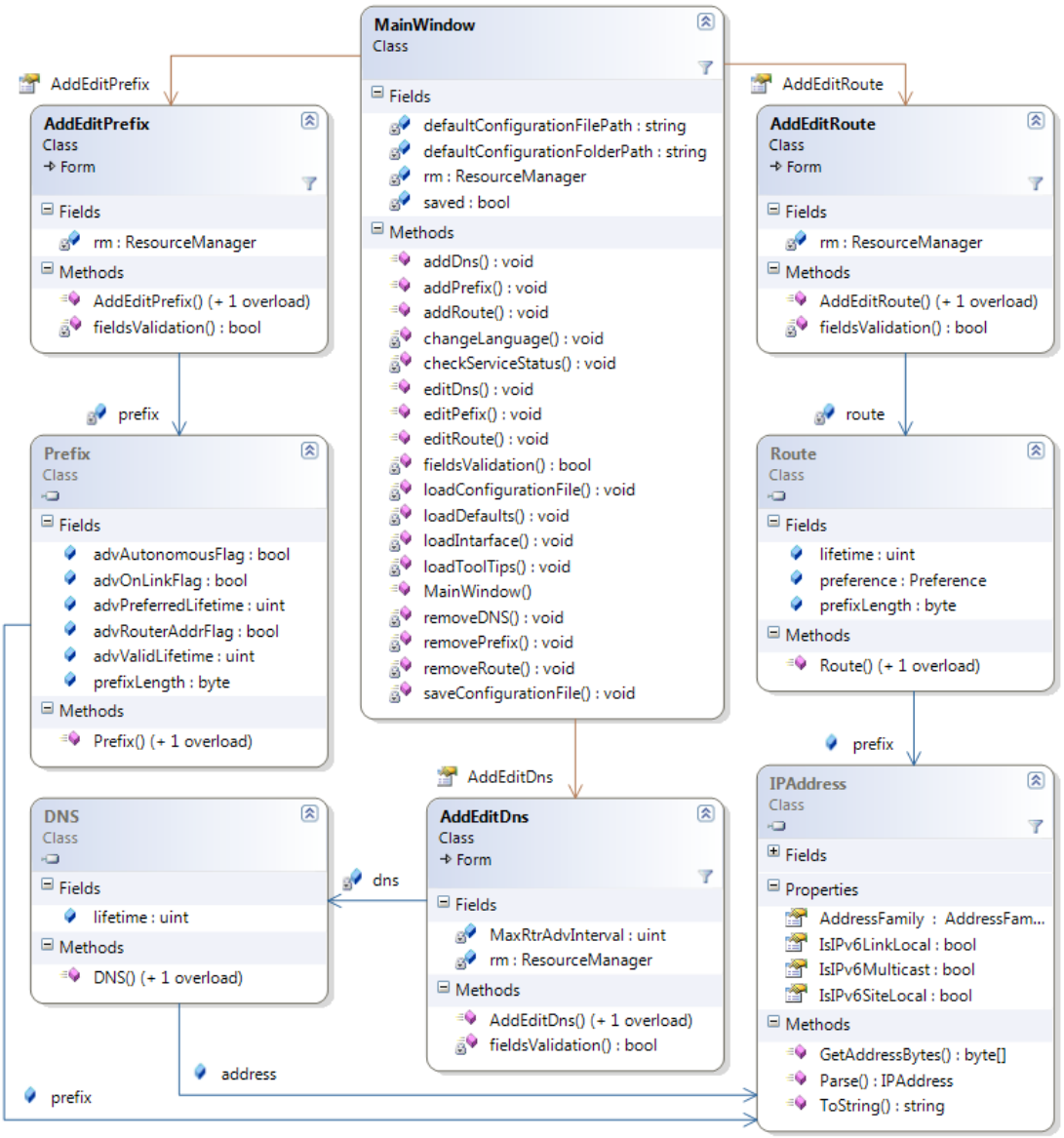

Figura 6.3: Diagrama de clases para la herramienta de configuración del servidor.

En la Figura 6.3 se observa como la clase *MainWindow* posee los métodos para realizar las funciones de la herramienta. Hay tres clases asociadas a la ventana *MainWindow* que son *AddEditPrefix*, *AddEditRoute* y *AddEditDns,* cada una de estas genera una ventana de tipo *ToolWindow* y sirven para aceptar los datos de entrada que llenarán las opciones que se anidaran en el mensaje RA a enviar por la interfaz seleccionada.

En el diagrama de la clase *Interface* (Figura 6.2) se puede observar que existen tres listas, una de ellas con objetos de tipo *Prefix*, otra con objetos de

tipo *Route* y por último una con objetos de tipo *DNS*. Cada una de esas clases representa las posibles opciones que se envían en los mensajes RA. Por lo tanto, los atributos de dichas clases son equivalentes a los parámetros de las opciones que se agregan a los mensajes RA si el usuario desea configurarlas.

En cada una de las listas antes mencionadas al menos uno de los parámetros es una dirección IPv6. Las direcciones se representan con un objeto de tipo *IPAddress*. Entre las propiedades más importantes de estos objetos se encuentra *AddressFamily* porque permite seleccionar sólo las direcciones IPv6. También se destacan *IsIPv6LinkLocal*, *IsIPv6Multicast* e *IsIPv6SiteLocal* porque ayudan a identificar el tipo de dirección IPv6 que se maneja.

Para culminar el *Sprint Backlog* se presenta la interfaz gráfica final de la herramienta de configuración del servidor que se puede observar en la Figura 6.4.

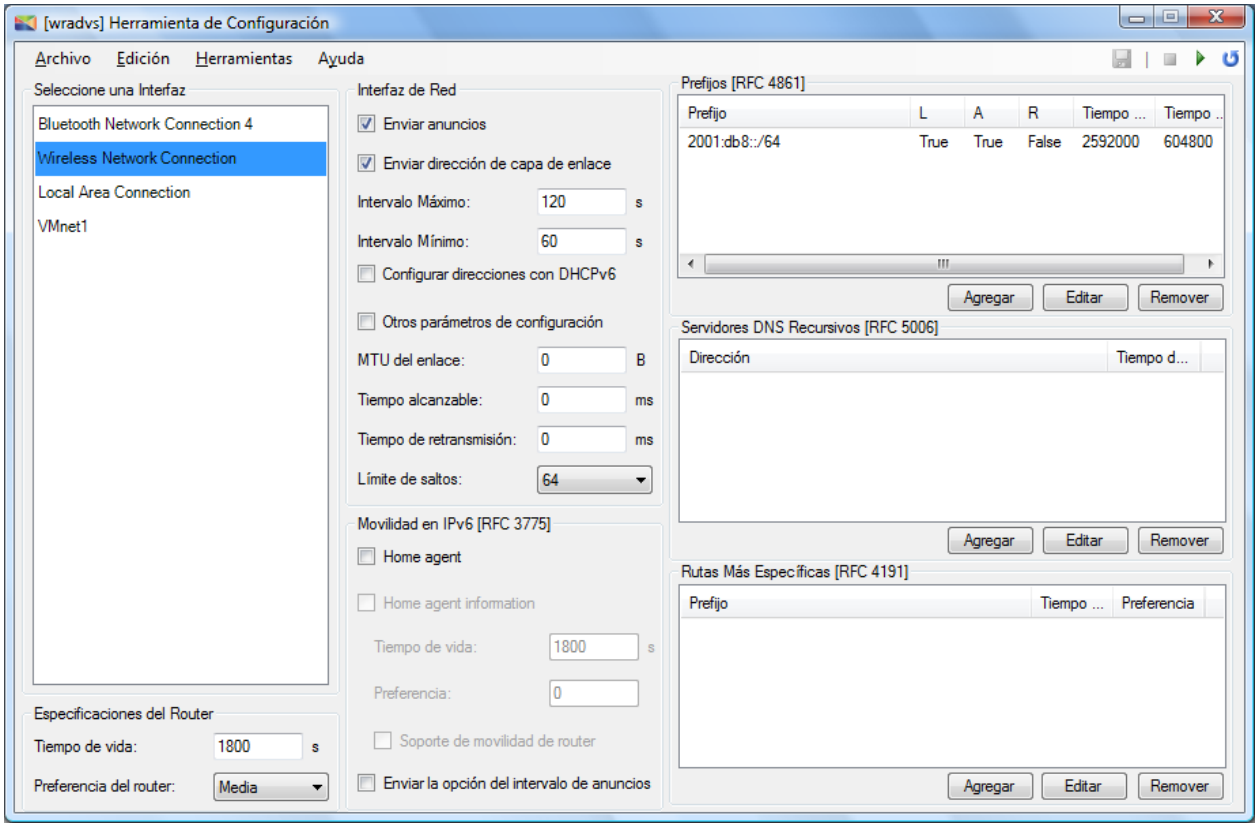

Figura 6.4: Interfaz gráfica de la herramienta de configuración del servidor.

6.3 Sprint Review

Para comprender mejor los procesos que ocurren en la herramienta de configuración del servidor se analizó el código de los métodos más relevantes. Estos son:

- loadConfigurationFile.
- saveConfigurationFile.
- startService.
- FieldsValidation.

El método *loadConfigurationFile* de la clase *MainWindow* se encarga de inicializar la lista de interfaces de red, para que cuando el usuario seleccione una de ellas se llenen los campos de la ventana con los valores existentes en el archivo de configuración (*configuration.xml*). Para cargar cada interfaz de red desde este archivo se utilizó un método de la clase *Util* del espacio de nombre *ClassLibrary* con el mismo nombre (*loadConfigurationFile*), que realiza la lectura del archivo XML. En la Figura 6.5 se muestra un fragmento del código utilizado para leer el archivo XML y así crear los objetos de tipo *Interface* para llenar la lista de interfaces que puede seleccionar el usuario.

```
1  public static IEnumerable<Interface> loadConfigurationFile(string path)
2  {
3      XElement xml = XElement.Load(path);
4      return from i in xml.Elements("Interface") select new Interface {
5      id = (string)i.Attribute("id") ?? "",
6      name = (string)i.Attribute("name") ?? "",
7      index = (int?)i.Attribute("index") ?? -1,
8      advSendAdvertisements = (bool?)i.Element("AdvSendAdvertisements") ?? false,
9      advSourceLLAddress = (bool?)i.Element("AdvSourceLLAddress") ?? true,
10     maxRtrAdvInterval = (UInt32)((int?)i.Element("MaxRtrAdvInterval") ?? 600),
11     minRtrAdvInterval = (UInt32)((int?)i.Element("MinRtrAdvInterval") ?? 198),
12     advManagedFlag = (bool?)i.Element("AdvManagedFlag") ?? false,
13     advOtherConfigFlag = (bool?)i.Element("AdvOtherConfigFlag") ?? false,
14     advHomeAgentFlag = (bool?)i.Element("AdvHomeAgentFlag") ?? false,
15     advDfltPreference = (Preference)((int?)i.Element("AdvDefaultPreference") ??
1),
16     advLinkMtu = (UInt32?)i.Element("AdvLinkMtu") ?? 0,
17     advReachableTime = (UInt32?)i.Element("AdvReachableTime") ?? 0,
18     advRetransTimer = (UInt32?)i.Element("AdvRetransTimer") ?? 0,
19     advCurHopLimit = (byte)((int?)i.Element("AdvCurHopLimit") ?? 64),
20     advDefaultLifetime = (UInt16)((int?)i.Element("AdvDefaultLifetime") ??
1800),
21     advHomeAgentInfo = (bool?)i.Element("AdvHomeAgentInfo") ?? false,
22     homeAgentLifetime = (UInt16)((int?)i.Element("HomeAgentLifetime") ?? 1800),
23     homeAgentPreference = (UInt16)((int?)i.Element("HomeAgentPreference") ?? 0),
24     advMobRtrSupportFlag = (bool?)i.Element("AdvMobRtrSupportFlag") ?? false,
25     advIntervalOpt = (bool?)i.Element("AdvIntervalOpt") ?? false,
26     nic = Util.getNetworkInterface(i.Attribute("id").Value ?? ""),
```

Figura 6.5: Fragmento de código del método *loadConfigurationFile* de la clase *Util*.

El fragmento de código de la Figura 6.5 muestra como es leído el archivo de configuración. En la línea 4 (Figura 6.5) se muestra el query utilizado para tomar cada elemento *Interface* del archivo XML (para ver un ejemplo de archivo de configuración ir a la Figura 5.5).

El resto de las líneas muestran como son llenados los parámetros de interfaz de los mensajes RA. Por ejemplo: en la línea 8 (Figura 6.5) se verifica si el valor del elemento *AdvSendAdvertisements* es *null* o no. Si el valor es *true* o *false* es asignado a la variable *advSendAdvertisements* del objeto *Interface*. Si el valor es *null* entonces es colocado el valor por defecto, es decir *false*. De esta forma si el archivo de configuración es modificado manualmente de forma incorrecta, se pueden prever los posibles errores.

El siguiente método mencionado es llamado *saveConfigurationFile*, su función es almacenar los cambios realizados por el usuario en el archivo de configuración, de modo que cuando el usuario decida iniciar el servicio se utilice el archivo actualizado. En la Figura 6.6 se puede observar un fragmento del código utilizado para llenar el archivo XML.

```
1  private void saveConfigurationFile(string path)
2  {
3      XmlTextWriter xml = new XmlTextWriter(path, null);
4      xml.Formatting = Formatting.Indented;
5      xml.WriteStartDocument();
6      xml.WriteStartElement("Interfaces");
7
8      for (int i = 0; i < interfacesList.Items.Count; i++)
9      {
10         Interface inter = (Interface)interfacesList.Items[i];
11
12         xml.WriteStartElement("Interface");
13         xml.WriteAttributeString("id", inter.id);
14         xml.WriteAttributeString("name", inter.name);
15         xml.WriteAttributeString("index", inter.index.ToString());
16         xml.WriteElementString("AdvSendAdvertisements",
17                                inter.advSendAdvertisements.ToString());
18         xml.WriteElementString("AdvSourceLLAddress",
19                                inter.advSourceLLAddress.ToString());
20         xml.WriteElementString("MaxRtrAdvInterval",
21                                inter.maxRtrAdvInterval.ToString());
22         xml.WriteElementString("MinRtrAdvInterval",
23                                inter.minRtrAdvInterval.ToString());
24         xml.WriteElementString("AdvManagedFlag",
inter.advManagedFlag.ToString());
25         xml.WriteElementString("AdvOtherConfigFlag",
26                                inter.advOtherConfigFlag.ToString());
```

Figura 6.6: Fragmento de código del método *saveConfigurationFile*.

En la línea 3 (Figura 6.6) se muestra el código utilizado para abrir el archivo XML indicado en el camino hacia el archivo (variable *path*). Este método se

encarga de salvar el archivo de configuración para que el servidor lo utilice, sin embargo, es posible guardar el archivo en otro directorio si el usuario lo desea, por lo tanto el camino no siempre será el mismo.

Las líneas 16 y 17 (Figura 6.6) muestran el comando necesario para que se escriba un valor en el elemento *AdvSendAdvertisments* del archivo XML. Asimismo, todos los campos son agregados uno a uno hasta que se obtiene el archivo de configuración final con la lista de las interfaces y sus estados.

El siguiente método relevante para esta herramienta es llamado *startService* y su función es iniciar el servicio. La Figura 6.7 muestra el código utilizado para realizar este proceso.

```
1  public static void startService(string serviceName, int timeoutMilliseconds,
2                                  string language)
3  {
4      ServiceController service = new ServiceController(serviceName);
5      try
6      {
7          int millisec1 = Environment.TickCount;
8          TimeSpan timeout = TimeSpan.FromMilliseconds(timeoutMilliseconds);
9          if (service.Status !=
10             System.ServiceProcess.ServiceControllerStatus.Running)
11         {
12             SplashScreen.SetStatus("Starting wradvs service");
13             service.Start();
14             service.WaitForStatus(ServiceControllerStatus.Running, timeout);
15         }
```

Figura 6.7: Fragmento de código del método *startService* de la clase *Util*.

Al iniciar el método se creó un objeto de tipo *ServiceController* que se llamó *service* (línea 4, Figura 6.7), este objeto es un hilo, que posee diferentes propiedades, una de ellas es el *Status*. Con esta propiedad se verifica si el hilo se encuentra corriendo para el momento en que se ha realizado la llamada al método (líneas 9 y 10, Figura 6.7). En caso de que el hilo no esté corriendo se ejecuta la instrucción *service.Start()* de la línea 13 (Figura 6.7), que no es más que indicarle al hilo *service* que debe iniciar su ejecución. Los métodos *stopService* y *restartService* funcionan de la misma forma que el método *startService*, la única diferencia se encuentra en el método que se ejecuta sobre el hilo. En la Figura 6.8 se muestra un fragmento del método *stopService* donde se detiene el proceso.

```
1  service.Stop();
2  service.WaitForStatus(ServiceControllerStatus.Stopped, timeout);
```

Figura 6.8: Fragmento de código del método *stopService* de la clase *Util*.

Por último, en el Sprint Review se explicó el método de validación de los campos al momento de mandar a iniciar la aplicación. Cada campo de la aplicación posee tres tipos de validación. El primer proceso es la verificación de la entrada de cada carácter de forma que el usuario sólo pueda introducir cifras o letras válidas (ver Figura 6.9, método *number_KeyPress*). El segundo proceso de verificación es realizado cuando el usuario coloca un nuevo carácter en el campo y se valida toda la cadena introducida hasta ese momento (ver ejemplo en el método *maxIntervalTextB_TextChange* de la Figura 6.10. El tercer método de verificación se realiza una vez que se desea guardar o iniciar el servidor, en ese punto se validan todos los campos para que la información sea congruente, en la Figura 6.11 se muestra un fragmento del código de del método *fieldsValidation*.

```
1  private void number_KeyPress(object sender, KeyPressEventArgs e)
2  {
3      tooltip.RemoveAll();
4      if (!System.Text.RegularExpressions.Regex.IsMatch(
5                                          e.KeyChar.ToString(), @"[\d\b\cC\cV]"))
6      {
7          tooltip.Show(MyToolTip.getMessage(ToolTipType.Numeric),
   MainWindow.FromHandle(((Control)sender).Handle), 0, -65, 5000);
8          e.Handled = true;
9      }
10 }
```

Figura 6.9: Código del método *number_KeyPress*.

En la Figura 6.9 se muestra el método *number_KeyPress* que es utilizado por todos los campos que sólo deben aceptar caracteres numéricos. Si el usuario intenta colocar algún otro carácter que no sean los dígitos del 0 al 9, se muestra un tooltip indicando al usuario que debe introducir solamente caracteres válidos.

```
1  private void maxIntervalTextB_TextChanged(object sender, EventArgs e)
2  {
3      if (maxIntervalTextBValidation(false))
4      {
5          maxIntervalTextB.BackColor = Color.White;
6          if (minIntervalTextBValidation(false))
7              minIntervalTextB.BackColor = Color.White;
8          else
9              minIntervalTextB.BackColor = Color.FromArgb(255, 128, 128);
10
11         if (routerLifetimeTextBValidation(false))
12             routerLifetimeTextB.BackColor = Color.White;
13         else
14             routerLifetimeTextB.BackColor = Color.FromArgb(255, 128, 128);
```

Figura 6.10: Fragmento de código del método *maxIntervalTextB_TextChanged*.

Las líneas 9 y 14 del método *maxIntervalTextB_TextChanged* (Figura 6.10) se encargan de colocar en color rojo los campos de la ventana en los que existen errores cuando se ha agregado o borrado un carácter en el campo *maxIntervalTextB*.

```
1  private bool fieldsValidation(bool showMsg)
2  {
3      bool validDNSLifetime = true;
4      bool validFields = true;
5
6      validFields = maxIntervalTextBValidation(showMsg) &&
7                    minIntervalTextBValidation(showMsg) &&
8                    reachableTimeTextBValidation(showMsg) &&
9                    retransTimerTextBValidation(showMsg) &&
10                   linkMtuTextBValidation(showMsg) &&
11                   routerLifetimeTextBValidation(showMsg) &&
12                   homePreferenceTextBValidation(showMsg) &&
13                   homeLifetimeTextBValidation(showMsg);
```

Figura 6.11: Fragmento de código del método *fieldsValidation*.

El último método mencionado es el *fieldsValidation*, que es llamado al momento de guardar el archivo de configuración y su función es validar cada uno de los campos de la ventana para verificar que se construya adecuadamente el archivo de configuración.

6.4 Sprint Retrospective

Durante esta etapa se realizaron los siguientes cambios:

- Se modificó el proceso de validación que inicialmente constaba de dos procesos.
- Se agregó el soporte para *Home Agent* y los servidores DNS recursivos.
- Se modificaron los tooltips iniciales para aclarar los nombres de las variables referentes a cada campo y se añadió la descripción.

7. Visor de Eventos

Este capítulo describe el *Sprint 3*. El visor de eventos permite visualizar el log de eventos del servidor en tiempo real, mostrando por cada interfaz de red, el valor de todos los campos de las cabeceras de los mensajes RA enviados y los mensajes RS recibidos.

7.1 Sprint Planning Meeting

Para la realización del módulo fue necesario definir los servicios que provee la herramienta, los cuales son mostrados en el diagrama de casos de uso de la Figura 7.1.

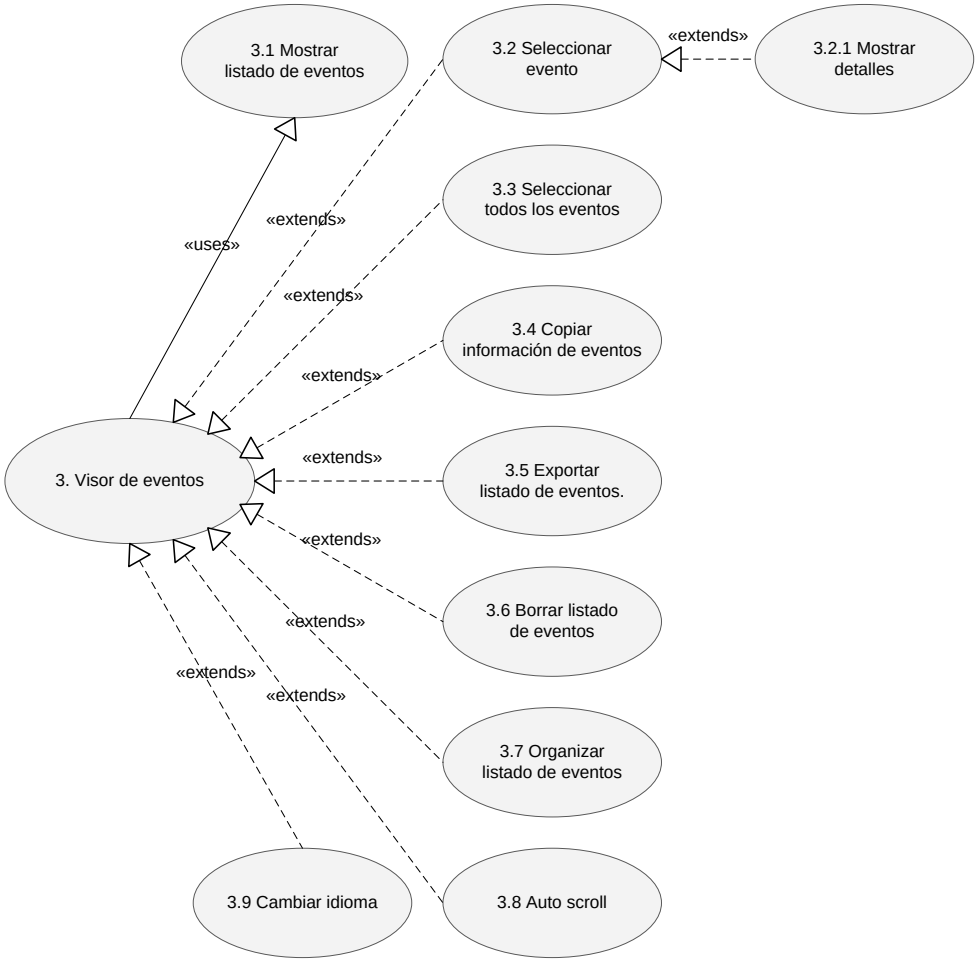

Figura 7.1: Diagrama de casos de uso 3 - nivel 1. Visor de eventos.

Los casos de uso son especificados en las tablas: Tabla 7.1, Tabla 7.2, Tabla 7.3, Tabla 7.4 Tabla 7.5 Tabla 7.6, Tabla 7.7, Tabla 7.9, Tabla 7.10.

Caso de uso
3.1 Mostrar listado de eventos
Actores
✓ Usuario.
Descripción
Se muestra un listado de todos los eventos producidos por el servidor.
Flujo básico
✓ Se cargan todos los eventos del log de eventos del servidor.

Tabla 7.1: Especificación de caso de uso 3.1 Mostrar listado de eventos.

Caso de uso
3.2 Seleccionar evento
Actores
✓ Usuario.
Descripción
Se selecciona un evento del listado.
Flujo básico
✓ Se hace clic en un elemento de la lista.

Tabla 7.2: Especificación de caso de uso 3.2 Seleccionar evento.

Caso de uso
3.2.1 Mostrar detalles
Actores
✓ Usuario.
Descripción
Se muestra una ventana con información detallada del evento seleccionado.
Flujo básico
✓ Se hace clic en un elemento de la lista. ✓ Se presiona el botón “Detalles” o se hace doble-clic en el elemento de la lista.

Tabla 7.3: Especificación de caso de uso 3.2.1 Mostrar detalles.

Caso de uso
3.3 Seleccionar todos los eventos
Actores
✓ Usuario.
Descripción
Se realiza selección de todos los eventos existentes en el listado.

Flujo básico
✓ Se presiona el botón “Seleccionar todo”.

Tabla 7.4: Especificación de caso de uso 3.3 Seleccionar todos los eventos.

Caso de uso
3.4 Copiar información de eventos
Actores
✓ Usuario.
Descripción
Se realiza una copia de la información de los eventos seleccionados.
Flujo básico
✓ Al presionar el botón “Copiar” se envía al portapapeles la información de los eventos seleccionados.

Tabla 7.5: Especificación de caso de uso 3.4 Copiar información de eventos.

Caso de uso
3.5 Exportar listado de eventos
Actores
✓ Usuario.
Descripción
Se exporta el listado de eventos a un documento de Excel (.xls).
Flujo básico
✓ Al presionar el botón “Exportar” se muestra un dialogo para guardar el documento de Excel (.xls).

Tabla 7.6: Especificación de caso de uso 3.5 Exportar listado de eventos.

Caso de uso
3.6 Borrar listado de eventos
Actores
✓ Usuario.
Descripción
Se eliminan todos los eventos del listado.
Flujo básico
✓ Al presionar el botón “Borrar Todo” se eliminan todas las entradas existentes en el log de eventos y del listado.

Tabla 7.7: Especificación de caso de uso 3.6 Borrar listado de eventos.

Caso de uso
3.7 Organizar listado de eventos
Actores
✓ Usuario.

Descripción
Se ordena el listado de eventos según el contenido de la columna deseada.
Flujo básico
✓ Al hacer clic en título de una columna, la tabla se ordenara de forma creciente o decreciente según el contenido de dicha columna.

Tabla 7.8: Especificación de caso de uso 3.7 Organizar listado de eventos.

Caso de uso
3.8 Auto scroll
Actores
✓ Usuario.
Descripción
Permite visualizar siempre el último evento producido por el servidor.
Flujo básico
✓ Activar/desactivar “Auto Scroll”. ✓ Si se encuentra activado, el scroll del listado se moverá automáticamente permitiendo visualizar siempre el último evento producido por el servidor.

Tabla 7.9: Especificación de caso de uso 3.8 Auto scroll.

Caso de uso
3.9 Cambiar idioma
Actores
✓ Usuario.
Descripción
Se ofrece al usuario un menú para modificar el lenguaje de la aplicación.
Flujo básico
✓ El usuario accede a la opción “Herramientas” -> “Idioma” y selecciona una de las dos opciones que se presentan (“English” o “Español”). ✓ Al hacer clic sobre la opción se reinicia la aplicación y utiliza ahora el idioma seleccionado.

Tabla 7.10: Especificación de caso de uso 3.9 Cambio de idioma.

7.2 Sprint Backlog

La interfaz gráfica del visor de eventos cuenta con dos ventanas, la ventana principal (*MainWindow*) y la ventana de detalles de los eventos (*EventDetail*). En el diagrama de clases de la Figura 7.2 se pueden observar las clases utilizadas:

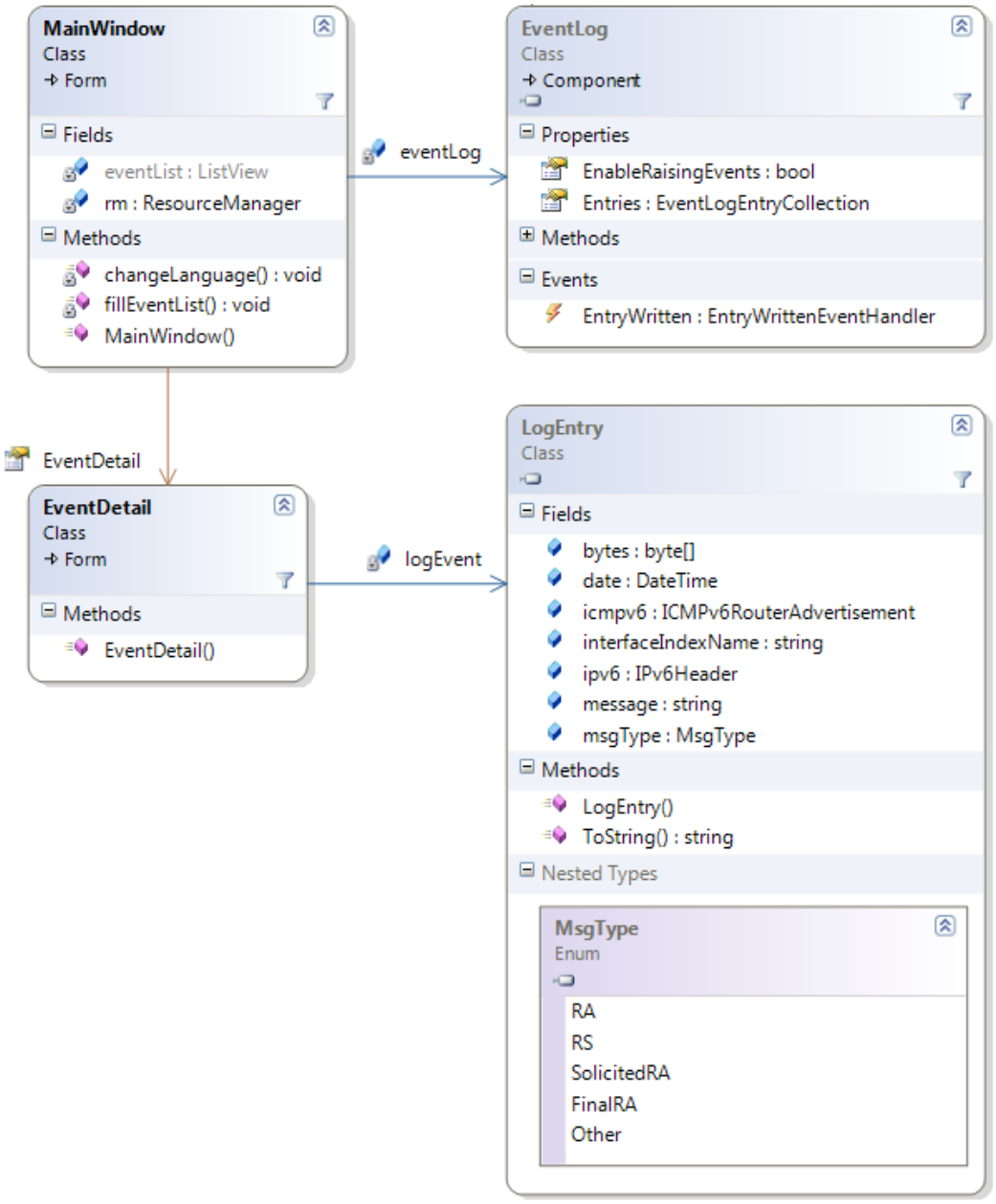

Figura 7.2: Diagrama de clases del Visor de eventos.

La ventana principal (*MainWindow*) posee una lista de eventos (*eventList*) del log del servidor, un objeto de tipo *EventLog* para acceder al log y obtener los eventos. Mediante el botón “Detalles” es posible acceder a una ventana de detalles de eventos (*EventDetail*), la cual posee un atributo de tipo *LogEntry*.

La interfaz gráfica de la ventana principal y de la ventana de detalles se puede observar en la Figura 7.3 y la Figura 7.4.

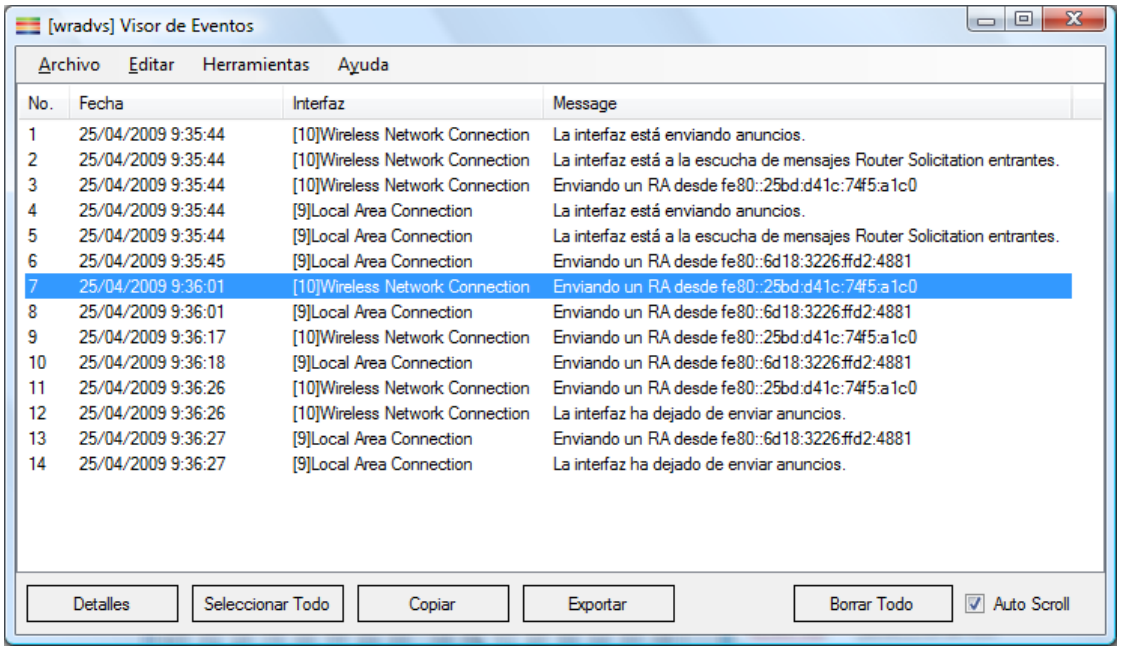

Figura 7.3: Ventana principal del visor de eventos.

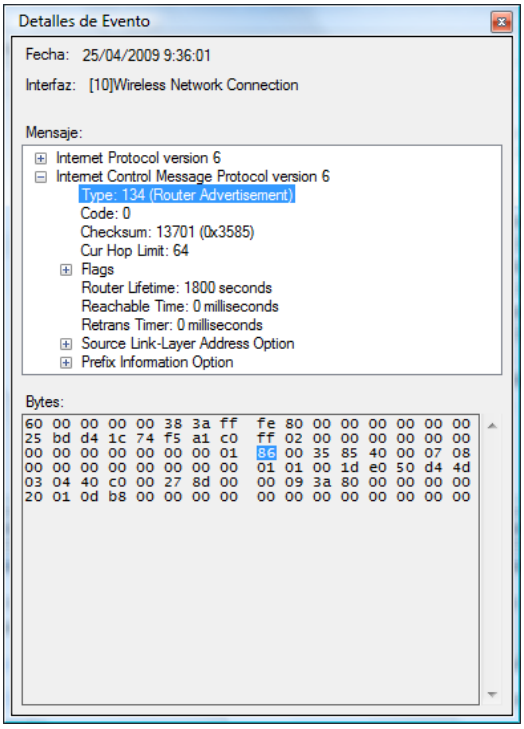

Figura 7.4: Ventana de detalles del visor de eventos.

La ventana de detalles de evento muestra la fecha en la que se produjo el evento y la interfaz relacionada al evento y en el caso de ser un mensaje RA

o RS, se muestran todos los valores de los campos de las cabeceras involucradas, incluyendo la cadena de bytes del mensaje.

7.3 Sprint Review

El acceso al log de eventos del servidor es logrado mediante la clase *EventLog*. La clase *MainWindow* del visor de eventos posee un atributo de este tipo, el cual es utilizado para llenar el listado de eventos. La Figura 7.5 muestra la inicialización de dicho atributo.

```
1  eventLog = new EventLog("wradvs log");
2  eventLog.EnableRaisingEvents = true;
```

Figura 7.5: Inicialización de atributo eventLog.

El constructor de la clase *EventLog* obtiene el nombre del log “wradvs log” (línea 1, Figura 7.5), el cual es el usado por el servidor para registrar los eventos. El llenado del listado es realizado mediante el método *fillEventList*, mostrado en la Figura 7.6.

```
1  private void fillEventList()
2  {
3      EventLogEntry[] entries = new EventLogEntry[eventLog.Entries.Count];
4      eventLog.Entries.CopyTo(entries, 0);
5      foreach (EventLogEntry entry in entries)
6      {
7          Util.LogEntry logEntry = new Util.LogEntry(entry);
8          ListViewItem item;
9          item = new ListViewItem((eventList.Items.Count+1).ToString());
10         item.SubItems.Add(logEntry.date.ToString());
11         item.SubItems.Add(logEntry.interfaceIndexName);
12         item.SubItems.Add(logEntry.ToString());
13         item.Tag = logEntry;
14         eventList.Items.Add(item);
15     }
16 }
```

Figura 7.6: Código del método fillEventList.

El primer paso fue obtener todos los eventos almacenados en el log; se puede observar en las líneas 3 y 4 de la Figura 7.6 la realización de una copia de la propiedad *entries* de *eventLog* a un arreglo de elementos del tipo *EventLogEntry*. El arreglo es recorrido con un ciclo *foreach* (línea 5, Figura 7.6), por cada elemento del arreglo, se crea un objeto de tipo *LogEntry* cuyo constructor recibe como parámetro el elemento del arreglo de la iteración en curso (línea 7, Figura 7.6). En las siguientes líneas se crea el ítem que será añadido al la lista *eventList*, el cual contiene la información de cada columna (número, fecha, nombre e índice de interfaz y el mensaje del evento).

El visor de eventos no sólo carga todos los eventos que se encuentran registrados en el log, sino que también muestra en tiempo real los eventos producidos por el servidor. Esto se realiza mediante un evento *EntryWritten* de la clase *EventLog*, el cual ejecuta un código cada vez que se registra un nuevo evento al log. A continuación se muestra dicho código (ver Figura 7.7).

```
1  private void eventLog_EntryWritten(object sender, EntryWrittenEventArgs e)
2  {
3      Util.LogEntry logEvent = new Util.LogEntry(e.Entry);
4      ListViewItem item = new ListViewItem((eventList.Items.Count+1).ToString());
5      item.SubItems.Add(logEvent.date.ToString());
6      item.SubItems.Add(logEvent.interfaceIndexName);
7      item.SubItems.Add(logEvent.ToString());
8      item.Tag = logEvent;
9      eventList.Items.Add(item);
10
11     if (autoScroll.Checked)
12         item.EnsureVisible();
13 }
```

Figura 7.7: Código del evento EntryWritten.

El procedimiento es igual al utilizado en el método *fillEventList* (desde la línea 3 hasta la línea 9, Figura 7.7). La línea 12 (Figura 7.7) se ejecuta cuando se encuentra seleccionada la casilla de activación “Auto Scroll” (ver Tabla 7.9), esta instrucción se encarga de hacer visible el ítem (evento) que acaba de ser agregado a la lista *eventList*.

El evento *EntryWritten* del objeto *eventLog* sólo se puede utilizar si se encuentra activada (valor *true*) la propiedad *EnableRaisingEvents* de dicho objeto (ver línea 2, Figura 7.5).

La clase *LogEntry* antes mencionada, se encarga de traducir el mensaje del evento creado por el servidor a una forma amigable para el usuario. Esta información es almacenada en los diferentes atributos de la clase (ver Figura 7.2).

La ventana de detalles de evento (clase *EventDetail*) es instanciada por medio del botón “Detalles” (ver Tabla 7.3). Esta clase recibe por parámetros un objeto de tipo *LogEntry*, del cual muestra toda la información que se posee (ver Figura 7.4).

7.4 Sprint Retrospective

- Se añadió la posibilidad de ordenar la lista de eventos según la columna deseada (ver Tabla 7.8).
- La ventana principal permite redimensionarse por el usuario.

8. Herramienta de Configuración de Direcciones IPv6

Este capítulo describe el *Sprint 4*. Esta herramienta proporciona a los usuarios una interfaz gráfica para realizar las funciones de la aplicación *netsh* relacionadas con la configuración de direcciones IPv6, incluyendo routers por defecto, administración de la tabla de enrutamiento, configuración de servidores DNS y activación de reenvío de paquetes (*forwarding*).

8.1 Sprint Planning Meeting

El *Sprint Goal* fue completar la herramienta de configuración de direcciones IPv6 en cuatro semanas. Para la realización de esta herramienta se analizaron diferentes formas de implementación. La manera más completa y que a su vez es compatible con Windows XP y sus sucesores fue realizando una interfaz gráfica para la aplicación *netsh*.

Las diferentes funcionalidades que puede ejecutar un usuario en está aplicación se ven reflejadas en el diagrama de casos de uso de la Figura 8.1:

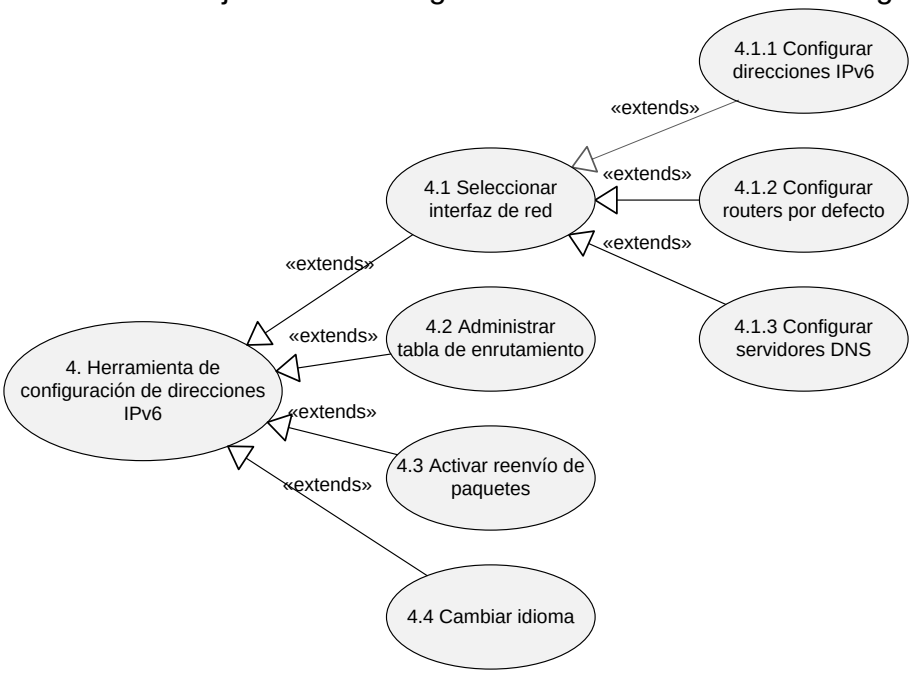

Figura 8.1: Diagrama de casos de uso 4 - nivel 1.
Herramienta de configuración de direcciones IPv6.

A continuación se muestra la especificación del nivel 1 de los casos de uso mencionados en la Figura 8.1, a través de las tablas: Tabla 8.2, Tabla 8.3, Tabla 8.4, Tabla 8.5, Tabla 8.6, Tabla 8.7.

Caso de uso
4.1 Seleccionar interfaz de red
Actores
✓ Usuario.
Descripción
Se selecciona una interfaz de red de la lista desplegable (“*combo box*”).
Flujo básico
✓ Se hace clic en un elemento de la lista desplegable.

Tabla 8.1: Especificación de caso de uso 4.1 Seleccionar interfaz de red.

Caso de uso
4.1.1 Configurar direcciones IPv6.
Actores
✓ Usuario.
Descripción
Se muestra una lista y tres botones para administrar las direcciones IPv6.
Flujo básico
✓ Si se presiona el botón “Agregar”, se muestra una ventana para introducir los datos necesarios para añadir una nueva dirección IPv6. ✓ Si se selecciona uno de los elementos de la lista y se presiona el botón “Editar” o se hace doble-clic, se muestra una ventana para modificar la información de la dirección IPv6 seleccionada. ✓ Si se selecciona un elemento de la lista y se presiona el botón “Remover”, se elimina la dirección IPv6 seleccionada.

Tabla 8.2: Especificación de caso de uso 4.1.1 Configurar direcciones IPv6.

Caso de uso
4.1.2 Configurar routers por defecto.
Actores
✓ Usuario.
Descripción
Se muestra una lista y tres botones para administrar las rutas por defecto.
Flujo básico
✓ Si se presiona el botón “Agregar”, se muestra una ventana para introducir los datos y añadir un nuevo router por defecto. ✓ Si se selecciona uno de los elementos de la lista y se presiona el botón “Editar” o se hace doble clic, se muestra una ventana para modificar la información de la dirección seleccionada. ✓ Si se selecciona un elemento de la lista y se presiona el botón “Remover”, éste es eliminado.

Tabla 8.3: Especificación de caso de uso 4.1.2 Configurar routers por defecto.

Caso de uso
4.1.3 Configurar servidores DNS.
Actores
✓ Usuario.
Descripción
Se muestra una lista y tres botones para administrar los servidores DNS.
Flujo básico
✓ Si se presiona el botón “Agregar”, se muestra una ventana para introducir los datos y añadir nuevo servidor DNS. ✓ Si se selecciona uno de los elementos de la lista y se presiona el botón “Editar” o se hace doble clic, se muestra una ventana para modificar la información del servidor DNS seleccionado. ✓ Si se selecciona un elemento de la lista y se presiona el botón “Remover”, se elimina el servidor DNS seleccionado.

Tabla 8.4: Especificación de caso de uso 4.1.3 Configurar servidores DNS.

Caso de uso
4.2 Administrar tabla de enrutamiento.
Actores
✓ Usuario.
Descripción
Se muestra una lista con todas las entradas de la tabla de enrutamiento y tres botones para administrarla.
Flujo básico
✓ Si se presiona el botón “Agregar”, se muestra un diálogo para añadir una entrada a la tabla de enrutamiento. ✓ Si se selecciona un elemento de la lista y se presiona el botón “Remover”, se elimina la ruta seleccionada.

Tabla 8.5: Especificación de caso de uso 4.2 Administrar la tabla de enrutamiento.

Caso de uso
4.3 Activar reenvío de paquetes.
Actores
✓ Usuario.
Descripción
Se muestra una lista de cuadros seleccionables (*checkbox*) para indicar en cuales interfaces de red se desea activar o desactivar el reenvío de paquetes.
Flujo básico
✓ Se muestra una lista de todas las interfaces de red IPv6 con un cuadro seleccionable a su lado para activar o desactivar el reenvío de paquetes.

Tabla 8.6: Especificación caso de uso 4.3 Activar reenvío de paquetes.

Caso de uso
4.4 Cambiar idioma
Actores
✓ Usuario.
Descripción
Se ofrece al usuario un menú para modificar el lenguaje de la aplicación.
Flujo básico
✓ El usuario accede a la opción “Herramientas” -> “Idioma” y selecciona una de las dos opciones que se presentan (“English” o “Español”). ✓ Al hacer clic sobre la opción se reinicia la aplicación y utiliza ahora el idioma seleccionado.

Tabla 8.7: Especificación de caso de uso 4.4 Cambio de idioma.

8.2 Sprint Backlog

Durante esta etapa del Sprint se realizó el código y se completó la interfaz gráfica de del herramienta de configuración de direcciones IPv6 basada en el prototipo propuesto. La interfaz consta de una ventana principal (*MainWindow*) con pestañas que separan las diferentes funcionalidades de la herramienta. En el diagrama de clases de la Figura 8.2 se puede observar la estructura de las clases.

La clase *MainWindow* posee la interfaz gráfica principal de la herramienta de configuración de direcciones IPv6. Cada funcionalidad de la herramienta posee una lista y diferentes botones que son agrupados en cajas de grupo (*GroupBox*).

Cuando las funciones no se relacionan directamente se colocan en diferentes pestañas (*Tabs*). En la Figura 8.3 se muestra la ventana principal de la herramienta de configuración de direcciones IPv6.

Las cuatro clases asociadas a la clase *MainWindow* son ventanas auxiliares de tipo *ToolWindow* utilizadas para colocar los datos de entrada que son procesados en la ventana.

La herramienta ‘realiza un llamado a la aplicación *netsh* y se mantiene corriendo en el fondo (en el *background*), de esta forma cada vez que se desea ejecutar un comando por consola, de esta forma la aplicación no es iniciada para cada instrucción permitiendo que la configuración sea mucho más eficiente.

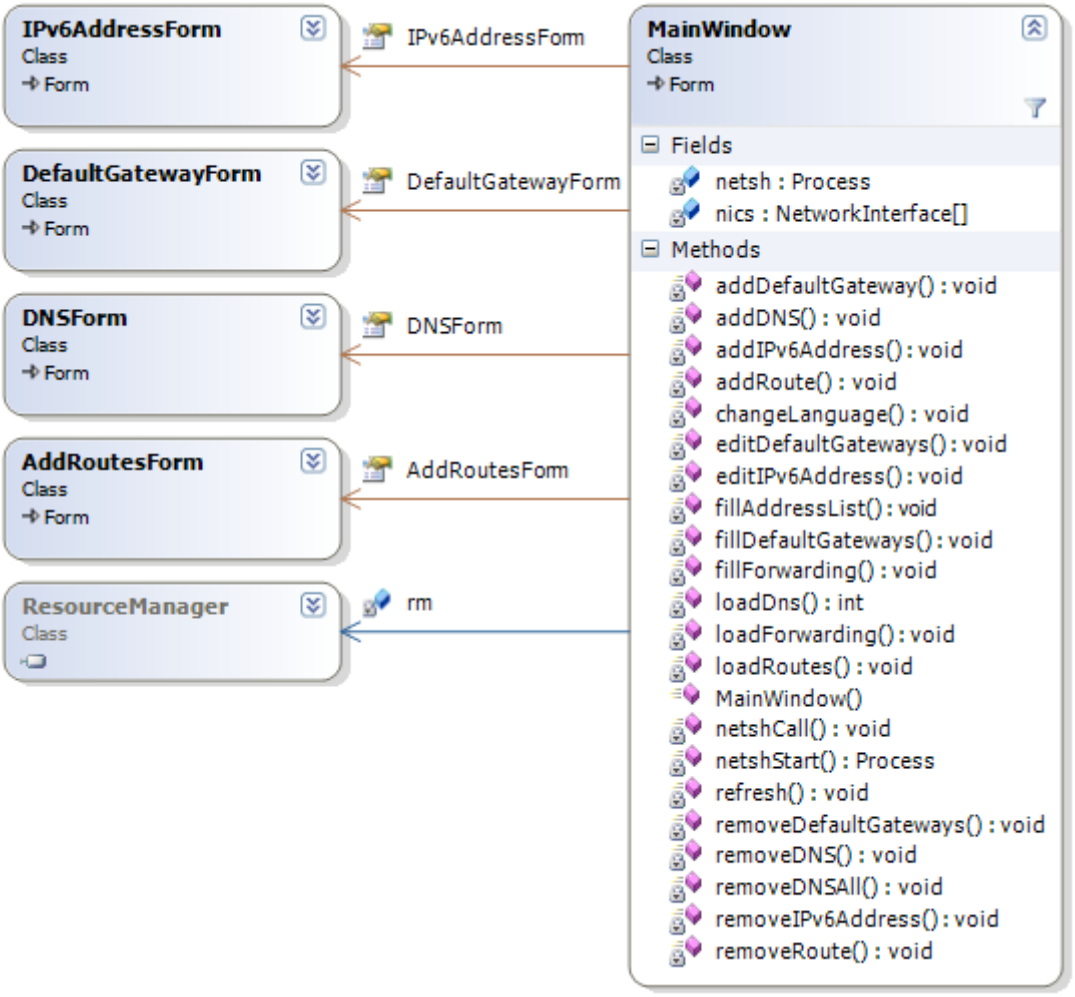

Figura 8.2: Diagrama de clases 4.
Herramienta de configuración de direcciones IPv6.

El atributo *netsh* de la clase *MainWindow* contiene el proceso a través de el cual se realizan los llamados a la aplicación *netsh*. Para iniciar el proceso se utilizó el método *netshStart(),* para ejecutar cada comando se utilizó el método *netshCall()*.

El resto de los métodos se encargan del llenado de las listas que se muestran por pantalla. Para hacerlo se toma la salida de cada ejecución, la cual se encuentra en una propiedad del proceso *netsh* llamada *StandardOutput.* Esta salida es procesada línea por línea para obtener los valores importantes para la aplicación.

Por último existe un botón “Recargar” que ejecuta el método *refresh()* que se encarga de actualizar todas las pestañas de forma que si se realiza algún cambio fuera de la aplicación este se refleje en ella.

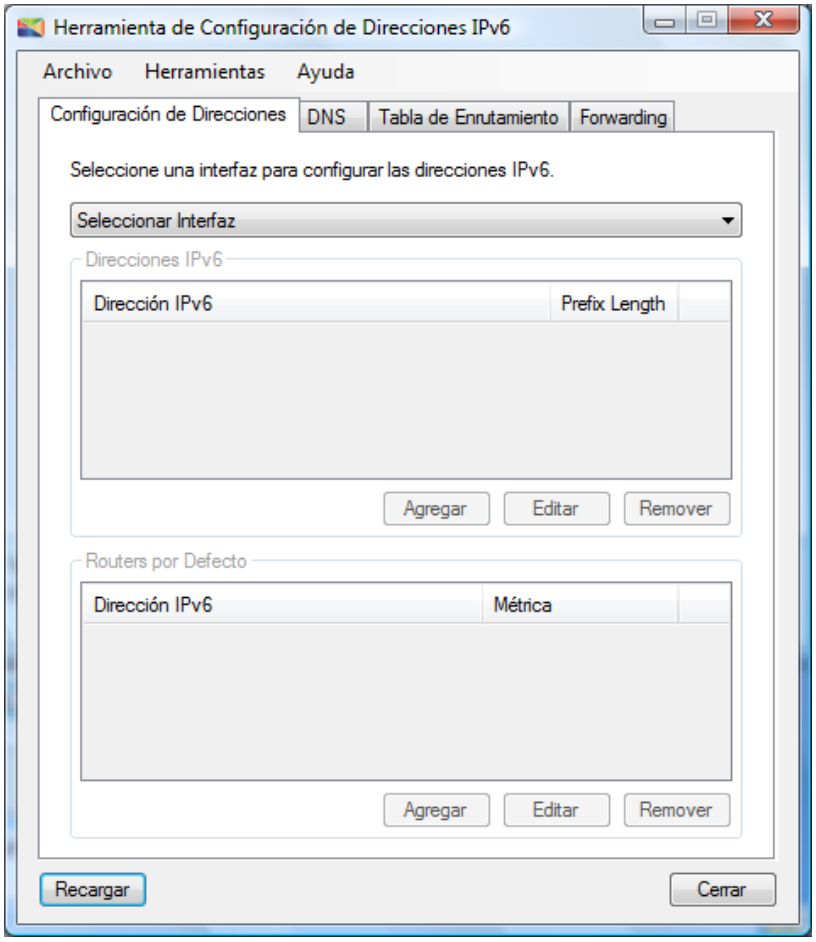

Figura 8.3: Ventana principal Herramienta de configuración de direcciones IPv6.

8.3 Sprint Review

Durante el Sprint Review se revisó el código fuente para analizar los métodos más importantes. Dentro de los aspectos más relevantes se destacan:

- ✓ El uso de la clase *System.Net.NetworkInformation.NetworkInterface* para la obtención de la información de las interfaces de red (identificador de interfaz, nombre de la interfaz y demás información relevante como direcciones IP). En la Figura 8.4 se muestra un fragmento de código con el cual se obtienen sólo las interfaces activas que poseen el stack de protocolos IPv6. Una vez obtenidas, las interfaces son colocadas en un *ComboBox* para que el usuario seleccione la interfaz de la lista y asi pueda obtener la información que requiere.

```
1
2  NetworkInterface[] nics;
3  nics = NetworkInterface.GetAllNetworkInterfaces();
4  comRoutesSelect.DropDownStyle = ComboBoxStyle.DropDownList;
5  comRoutesSelect.Items.Add("Select an interface");
6  foreach (NetworkInterface nic in nics)
7      if (nic.Supports(NetworkInterfaceComponent.IPv6))
8          if (!comRoutesSelect.Items.Contains(nic.Name))
9          {
10             comRoutesSelect.Items.Add(nic.Name);
11         }
```

Figura 8.4: Fragmento de código para la extracción de las interfaces de red IPv6.

En la línea 3 de la Figura 8.4 se inicializó la variable *nics* de tipo *NetworkInterface*. Esta variable posee todas las interfaces de red activas, incluyendo aquellas que sólo funcionan con IPv4.

En la línea 7 (Figura 8.4) se seleccionan sólo aquellas interfaces que poseen IPv6, excluyendo así las interfaces que únicamente poseen IPv4.

Por último la línea *comRoutesSelect.Items.Add(nic.Name)* en la línea 10 (Figura 8.4) llena una lista con los nombres de las interfaces para que el usuario seleccione la interfaz que desea configurar.

✓ Dado que la herramienta de configuración de direcciones IPv6 provee una interfaz gráfica para la aplicación *netsh* es importante hacer que el acceso a ésta sea lo más rápido posible. Por esta razón se creó un método llamado *netshStart* cuya función es iniciar un hilo que mantiene la aplicación a la espera de los comandos en un objeto de tipo *Process*. El método *netshStart* se puede observar en la Figura 8.5 a continuación:

```
1  private System.Diagnostics.Process netshStart()
2  {
3      System.Diagnostics.Process process = new System.Diagnostics.Process();
4      process.EnableRaisingEvents = false;
5      process.StartInfo.FileName = “netsh”;
6
7      process.StartInfo.UseShellExecute = false;
8      process.StartInfo.RedirectStandardInput = true;
9      process.StartInfo.RedirectStandardOutput = true;
10     process.StartInfo.CreateNoWindow = true;
11     process.Start();
12
13     process.StandardInput.WriteLine("interface ipv6");
14
15     return process;
16 }
```

Figura 8.5: Código del método netshStart.

La línea 5 (Figura 8.5) se utilizó para indicar al sistema que debe iniciar la aplicación *netsh*. En la línea 11 (Figura 8.5) con la instrucción *process.Start()* se activó el hilo para que la aplicación funcione como un demonio a la espera de futuros comandos. Con la línea 13 (Figura 8.5) se accede al modo *"interface ipv6"* de *netsh*. Por último, se retorna el proceso activo que es asignado a una variable global de forma que cada vez que sea necesario realizar un comando pueda realizarse sin tener que iniciar nuevamente la aplicación.

✓ El siguiente método importante para la herramienta es denominado *netshCall* y su función es llevar a cabo los comandos de *netsh* que permiten la configuración del sistema. Cada vez que se realiza una llamada al *netsh* la salida es almacenada en una propiedad de la clase *Process* llamada *StandardOutput*. El método *netshCall* se puede observar en la Figura 8.6:

```
1  private void netshCall(string command)
2  {
3      netsh.StandardInput.WriteLine(toASCII(command));
4  }
```

Figura 8.6: Código del método netshCall

La línea que compone este método se encarga de transformar el parámetro *command* en caracteres *ASCII* de modo que si hay acentos o caracteres especiales no falle la llamada al sistema. La propiedad *StandardInput* se accede desde el atributo *netsh* de tipo *Process* y se utilizó para ejecutar la nueva instrucción en la línea de comandos.

✓ El constructor de la ventana *MainWindow* que se muestra en la Figura 8.7, se encarga cargar el idioma de la aplicación y de inicializar el atributo *netsh*.

```
1  public MainWindow()
2  {
3      string lang = Util.getLanguage();
4      //loading the culture saved from the last use of the application
5      Thread.CurrentThread.CurrentUICulture=
   CultureInfo.CreateSpecificCulture(lang);
6      Thread.CurrentThread.CurrentCulture =
   CultureInfo.CreateSpecificCulture(lang);
7      InitializeComponent();
8
9      rm = new ResourceManager("IPv6AddressConfigurationTool.MainWindow",
10                              typeof(MainWindow).Assembly);
11
12     netsh = netshStart();
13 }
```

Figura 8.7: Código de cambio de idioma.

En la línea 3 (Figura 8.7) se utilizó el método *getLanguage()* de la clase *Util* perteneciente al espacio de nombre *ClassLibrary*. Este método lee un archivo de texto llamado *lang* que posee el idioma de la aplicación (“en” para inglés o “es” para español). Con la información del idioma se configura el *CurrentCulture* que indica que archivo de tipo *Resource* debe ser accedido. Para acceder a este archivo de idiomas se debe crear un objeto de tipo *ResourceManager* que maneja los archivos de tipo *Resource*.

Al finalizar los pasos antes mencionados el objeto *rm* de tipo *ResourceManager* posee acceso a los archivos XML que contienen las posibles traducciones, en la Figura 8.8 y la Figura 8.9 se muestra un fragmento de los archivos de configuración de idioma.

```
1  <data name="butAdvAddgateway.Text" xml:space="preserve">
2    <value>Add</value>
3  </data>
4  <data name="butAdvEditGateways.Text" xml:space="preserve">
5    <value>Edit</value>
6  </data>
```

Figura 8.8: Ejemplo del formato del archivo MainWindow.en.resx

```
1  <data name="butAdvAddgateway.Text" xml:space="preserve">
2    <value>Agregar</value>
3  </data>
4  <data name="butAdvEditGateways.Text" xml:space="preserve">
5    <value>Editar</value>
6  </data>
```

Figura 8.9: Ejemplo del formato del archivo MainWindow.es.resx

8.4 Sprint Retrospective

- Se agregó un botón para actualizar la ventana cuando el usuario lo desee de modo que si se realiza algún cambio en la configuración del sistema utilizando otro programa, los cambios se puedan ver reflejados en la herramienta.
- Se mejoró el proceso de acceso a *netsh* haciendo que este corra como un demonio mientras la aplicación esta activa.

9. Tooltips

Este capítulo describe el *Sprint 5*. Los tooltips son una herramienta cuya funcionalidad principal es que la aplicación sea utilizada con fines de enseñanza. Los tooltips deben ser objetivos y completos. Pueden contener definiciones, instrucciones de uso o información de la herramienta.

9.1 Sprint Planning Meeting

Durante la planificación se estudió la información que debía ser colocada en los diferentes campos de la herramienta de configuración del servidor. La información de los tooltips está basada en los diferentes RFCs mencionados en el capítulo 6 (herramienta de configuración del servidor).

9.2 Sprint Backlog

Para la creación de los tooltips se realizó una clase llamada *MyToolTip* que hereda de la clase *ToolTip* de C#. En la Figura 9.1 se muestra el diagrama de clases que representa la estructura utilizada para la creación de los tooltips.

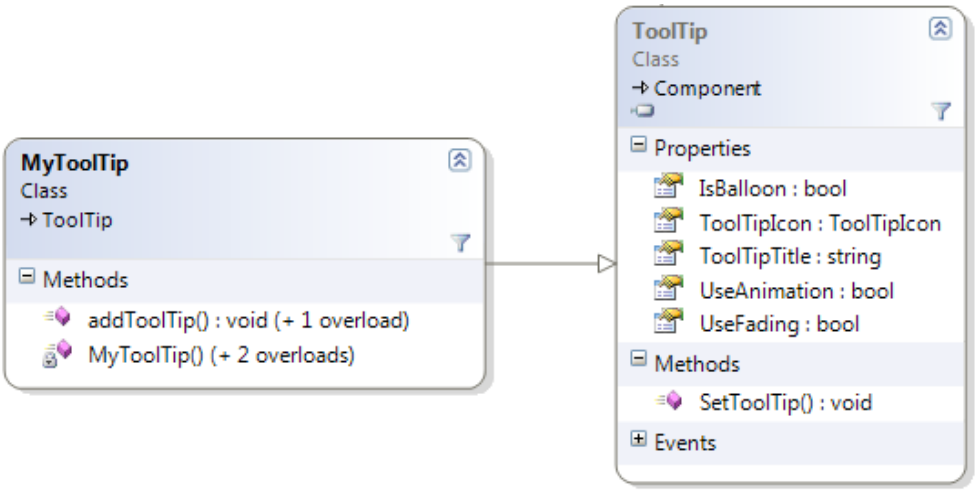

Figura 9.1: Diagrama de clases. MyToolTip.

La clase *MyToolTip* posee dos constructores y dos métodos llamados *addToolTip*. Cada constructor se encarga de darle formato al tooltip. Los métodos *addToolTip* crean el objeto y utilizan la función *setToolTip* de la clase *ToolTip* para mostrarlos por pantalla.

9.3 Sprint Review

En la Figura 2.1 se muestra un fragmento del método *loadTollTips* utilizado para cargar los tooltips en la ventana principal de la herramienta de configuración del servidor.

```
1   private void loadToolTips()
2   {
3       MyToolTip.addToolTip(sendAdvCheckB,
4           rm.GetString("sendAdvCheckBToolTip",
5           Thread.CurrentThread.CurrentUICulture), "AdvSendAdvertisements");
6       MyToolTip.addToolTip(sendSourceLLCheckbox,
7           rm.GetString("sendSourceLLCheckboxToolTip",
8           Thread.CurrentThread.CurrentUICulture), "AdvSourceLLAddress");
9       MyToolTip.addToolTip(maxIntervalLabel,
10          rm.GetString("maxIntervalLabelToolTip",
11          Thread.CurrentThread.CurrentUICulture), "MaxRtrAdvInterval");
12      MyToolTip.addToolTip(minIntervalLabel,
13          rm.GetString("minIntervalLabelToolTip",
14          Thread.CurrentThread.CurrentUICulture), "MinRtrAdvInterval");
```

Figura 9.2: Fragmento de código del método LoadToolTips

El primer parámetro de cada llamada al método *addToolTip* es de tipo *Control* y posee el objeto sobre el cual se debe ubicar el cursor del ratón para que el tooltip sea mostrado. El segundo parámetro es el texto que debe ser colocado en el tooltip. Dado que se soportan diferentes idiomas, para acceder a este texto, se utiliza el objeto *rm* de tipo *ResourceManager* que se encarga de acceder a la variable correspondiente al parámetro en el archivo XML del idioma actual de la aplicación. El idioma de la aplicación es conocido en la propiedad *Thread.CurrentThread.CurrentUICulture*.

9.4 Sprint Retrospective

Durante el Sprint Retrospective se llevaron a cabo correcciones en los textos de los tooltips y se agregaron los tooltips referentes a *Home Agent* y servidores DNS recursivos.

10. Pruebas

Este capítulo describe el *Sprint 6*. Durante la fase de pruebas se define una topología sobre la cual se analiza la reacción de diferentes sistemas operativos ante el envío de mensajes RA desde varios equipos actuando como servidores de autoconfiguración.

10.1 Sprint Planning Meeting

Las pruebas tienen como objetivo verificar que el servidor y sus herramientas funcionen correctamente.

Se diseñó una topología de tres hosts (A, B y C) y dos routers (D y E) para realizar las pruebas. La topología se muestra a acontinuación en la Figura 10.1:

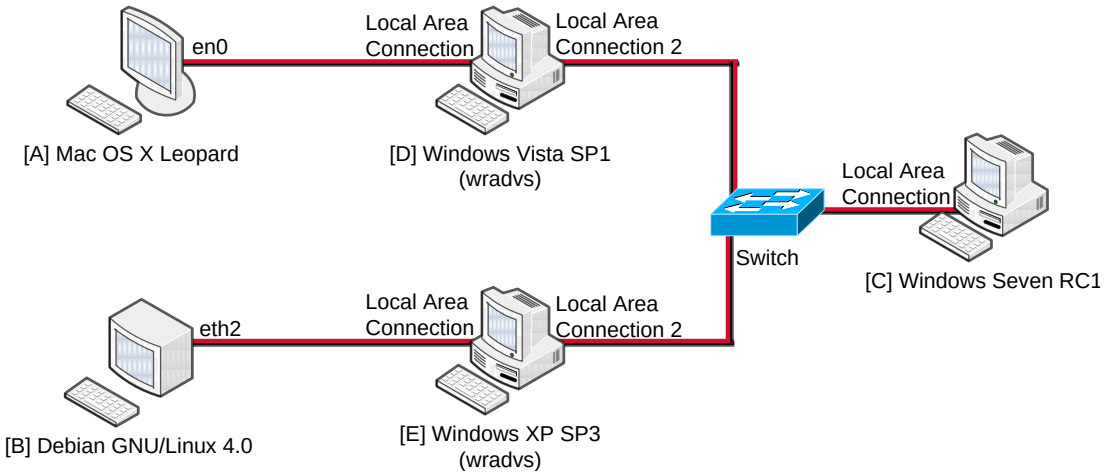

Figura 10.1: Topología de pruebas.

Para probar la interoperabilidad del servidor se utilizaron diferentes sistemas operativos:

- Mac OS X Leopard.
- Debian GNU/Linux 4.0.
- Windows XP Service Pack 3.
- Windows Vista Service Pack 1.
- Windows Seven Release Candidate 1

10.2 Sprint Backlog

10.2.1 Estado inicial

A continuación se detalla el estado inicial de los nodos de la red:

✓ **Host A**

El host A utiliza el sistema operativo Mac OS X Leopard. Posee una interfaz de red Ethernet llamada *en0* la cual tiene la dirección link-local fe80::21d:30ff:fe50:d4ae/64, derivada de la dirección de capa de enlace (ver Figura 10.2).

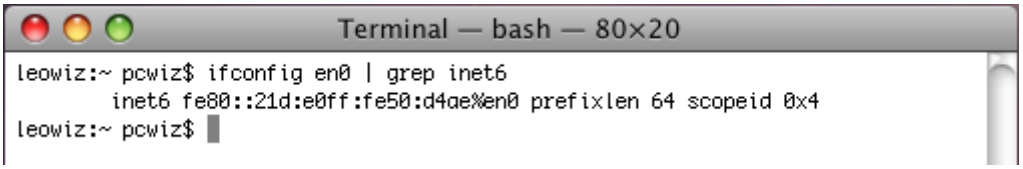

Figura 10.2: Estado inicial de la interfaz *en0* del host A.

Se utilizó el comando *"ifconfig en0 | grep inet6",* donde *"ifconfig en0"* se utiliza para obtener información de la interfaz de red *en0* y *"| grep inet6"* es utilizado para mostrar sólo las líneas que poseen la cadena *"inet6"*, logrando así enfocar sólo en la información relacionada a IPv6.

✓ **Host B**

El host B utiliza el sistema operativo Debian GNU/Linux 4.0. Posee una interfaz de red Ethernet llamada *eth2* la cual tiene la dirección link-local fe80::20c:29ff:fe74fc2b/64, derivada de la dirección de capa de enlace (ver Figura 10.3).

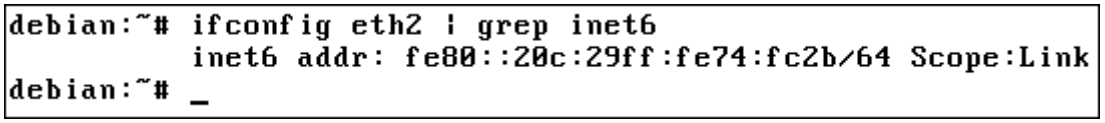

Figura 10.3: Estado inicial de la interfaz *eth2* del host B.

En este caso se utilizó el comando *"ifconfig eth3 | grep inet6"* dado que la interfaz lleva el nombre *eth2*. De igual forma se muestra sólo la información relacionada a IPv6.

✓ **Host C**

El host C utiliza el sistema operativo Windows Seven RC1. Posee una interfaz de red Ethernet llamada *Local Area Connection* la cual tiene la dirección link-local fe80::20c:29ff:fe47:8269/64 (ver Figura 10.4).

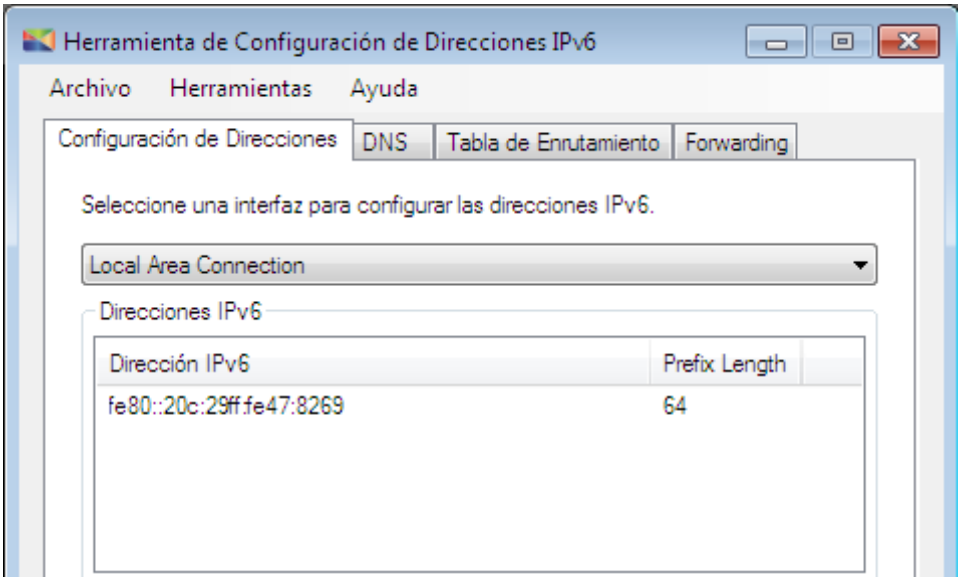

Figura 10.4: Estado inicial de la interfaz *Local Area Connection* del host C.

✓ **Router D**

El router D utiliza el sistema operativo Windows Vista SP1. Posee dos interfaces de red Ethernet:

- La interfaz *Local Area Connection* tiene la dirección link-local fe80::250:56ff:fec0:1/64 (ver Figura 10.5).

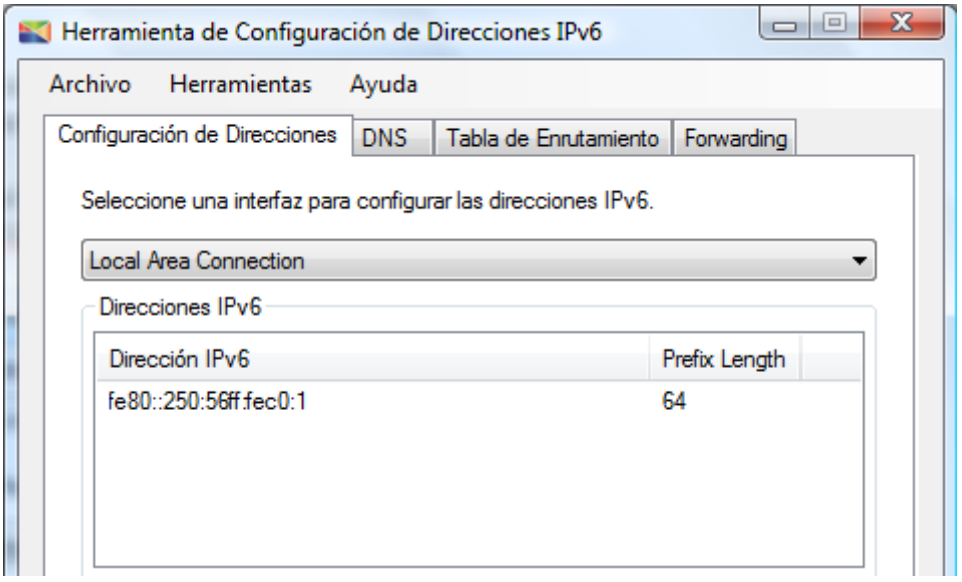

Figura 10.5: Estado inicial de la interfaz *Local Area Connection* del router D.

- La interfaz *Local Area Connection 2* tiene la dirección link-local fe80::215:c5ff:fe7f:bc10/64 (ver Figura 10.6)

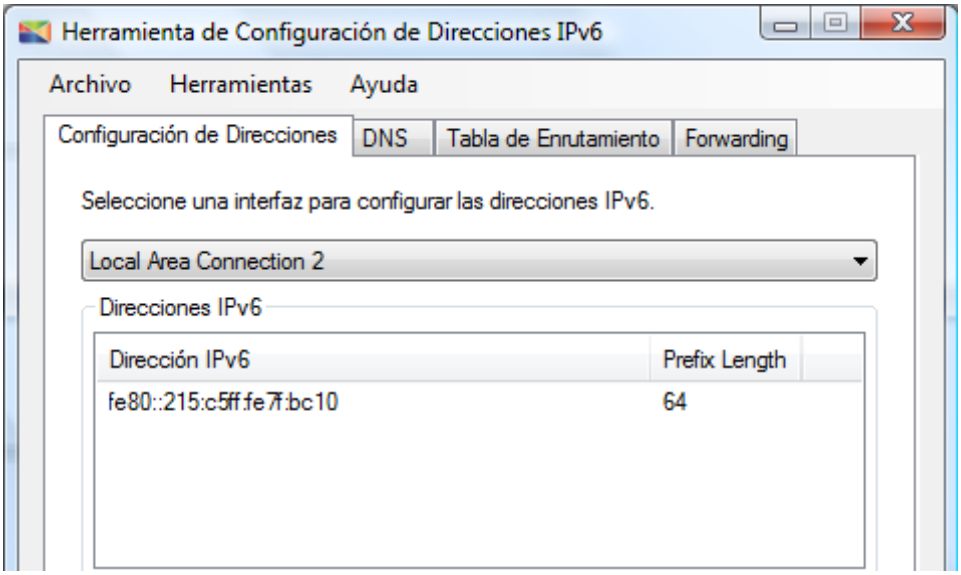

Figura 10.6: Estado inicial del de la interfaz *Local Area Connection 2* del router D.

✓ **Router E**

El router E utiliza el sistema operativo Windows XP SP3. Posee dos interfaces de red Ethernet:

- La interfaz *Local Area Connection* tiene la dirección link-local fe80::250:56ff:fec0:3/64 (ver Figura 10.7)

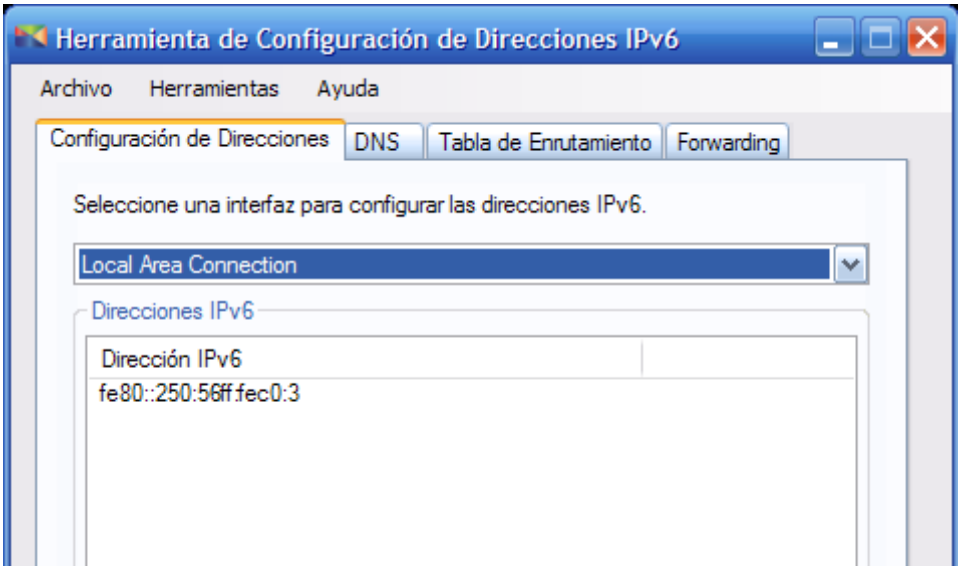

Figura 10.7: Estado inicial de la interfaz *Local Area Connection* del router E.

- La interfaz *Local Area Connection* 2 tiene la dirección link-local fe80::216:36ff:fe80:e7a4/64 (ver Figura 10.8).

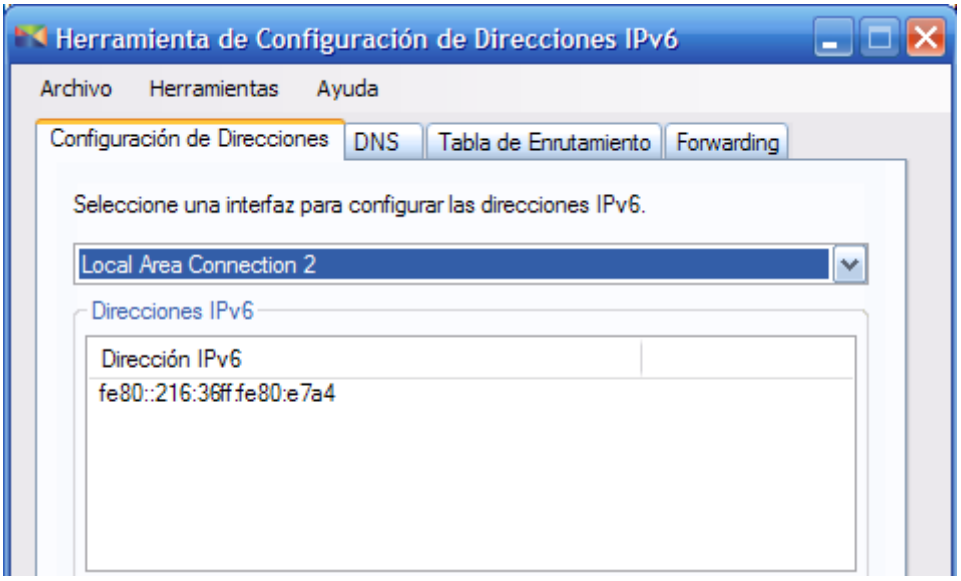

Figura 10.8: Estado inicial de la interfaz *Local Area Connection 2* del router E.

10.2.2 Configuración de los routers

Los routers E y D necesitan ser configurados para anunciar mensajes RA y comportarse como routers por defecto. A continuación se muestra la configuración realizada en los routers:

✓ **Router D**

Las dos interfaces de red del router D necesitan configurarse apropiadamente (ver Figura 10.9 y Figura 10.10):

- **Local Area Connection**

Se configuraron los siguientes parámetros de interfaz:

- **Enviar anuncios:** se selecciona la casilla para activar el envío de mensajes RA.
- **Intervalo máximo:** se le asigna 120 segundos, indicando que los mensajes RA deben ser anunciados en un intervalo de tiempo no mayor a dos minutos.
- **Intervalo mínimo:** se le asigna 60 segundos, indicando que los mensajes RA deben ser anunciados en un intervalo de tiempo no menor a un minuto.

Se configuraron los siguientes parámetros de especificaciones del router:

- **Tiempo de vida**: 1800 segundos, indicando que los hosts que reciben el mensaje RA deberán configurar al router D como router por defecto.
- **Preferencia del router**: media.

Se configuró el anuncio del siguiente prefijo:

- **Prefijo:** 2001:db8:1::/64.
- **Dentro del enlace:** sí, el prefijo se encuentra dentro del enlace.
- **Autoconfiguración sin estado:** sí, se deben generar direcciones a partir del prefijo.
- **Tiempo de vida:** 300 segundos, indicando que el prefijo será válido para determinación de rutas por 5 minutos.
- **Tiempo preferido:** 240 segundos, indicando que las direcciones generadas a partir del prefijo permanecerán en estado “preferido” por 4 minutos.

Se configuró el anuncio de las siguientes rutas más específicas:

- **Prefijo:** 2001:db8::/64.
- **Tiempo de vida:** 360 segundos, indicando que el prefijo será utilizado en determinación de rutas durante 6 minutos.
- **Preferencia:** media.

- **Prefijo:** 2001:db8:2::/64.
- **Tiempo de vida:** 360 segundos, indicando que el prefijo será utilizado en determinación de rutas durante 6 minutos.
- **Preferencia:** media.

- **Local Area Connection 2**

Se configuraron los siguientes parámetros de interfaz:

- **Enviar anuncios:** se selecciona la casilla para activar el envío de mensajes RA.
- **Intervalo máximo:** se le asigna 180 segundos, indicando que los mensajes RA deben ser anunciados en un intervalo de tiempo no mayor a tres minutos.
- **Intervalo mínimo:** se le asigna 120 segundos, indicando que los mensajes RA deben ser anunciados en un intervalo de tiempo no menor a dos minuto.

Se configuraron los siguientes parámetros de especificaciones del router:

- **Tiempo de vida**: 1800 segundos, indicando que los hosts que reciben el mensaje RA deberán configurar al router D como router por defecto.
- **Preferencia del router**: alta.

Se configuró el anuncio del siguiente prefijo:

- **Prefijo:** 2001:db8::/64.
- **Dentro del enlace:** sí, el prefijo se encuentra dentro del enlace.
- **Autoconfiguración sin estado:** sí, se deben generar direcciones a partir del prefijo.
- **Tiempo de vida:** 300 segundos, indicando que el prefijo será válido para determinación de rutas por 5 minutos.
- **Tiempo preferido:** 240 segundos, indicando que las direcciones generadas a partir del prefijo permanecerán en estado “preferido” por 4 minutos.

Se configuró el anuncio de la siguiente ruta más-específica:

- **Prefijo:** 2001:db8:1::/64.
- **Tiempo de vida:** 540 segundos, indicando que el prefijo será utilizado en determinación de rutas durante 9 minutos.
- **Preferencia:** media.

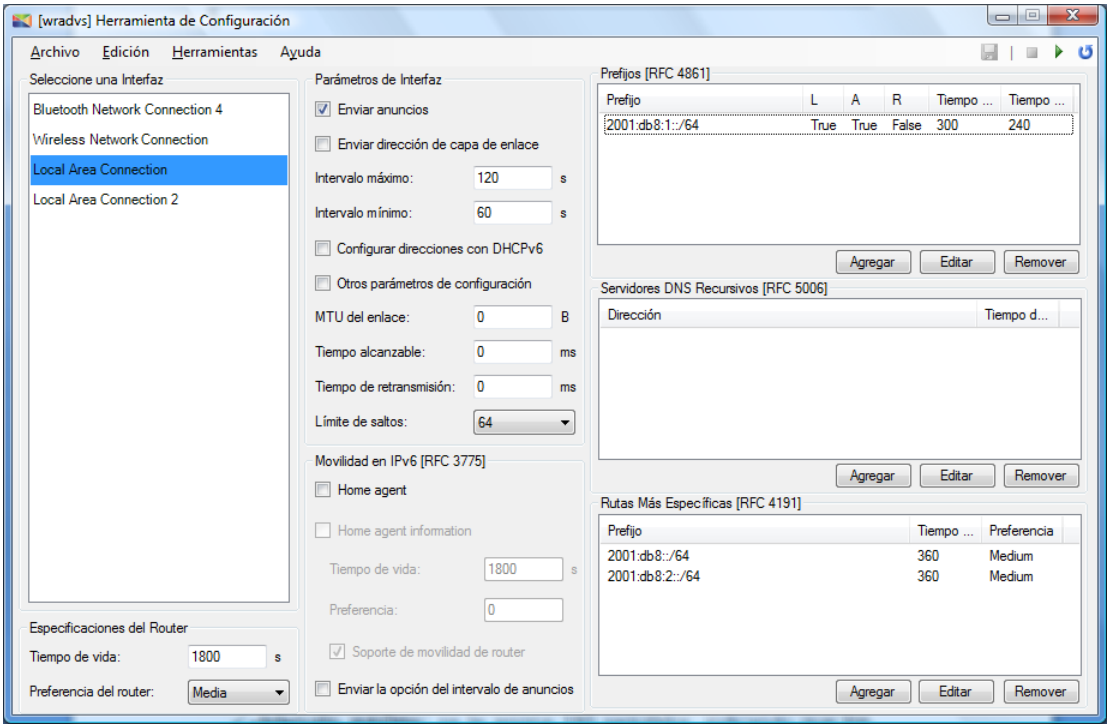

Figura 10.9: Configuración de la interfaz *Local Area Connection* del router D.

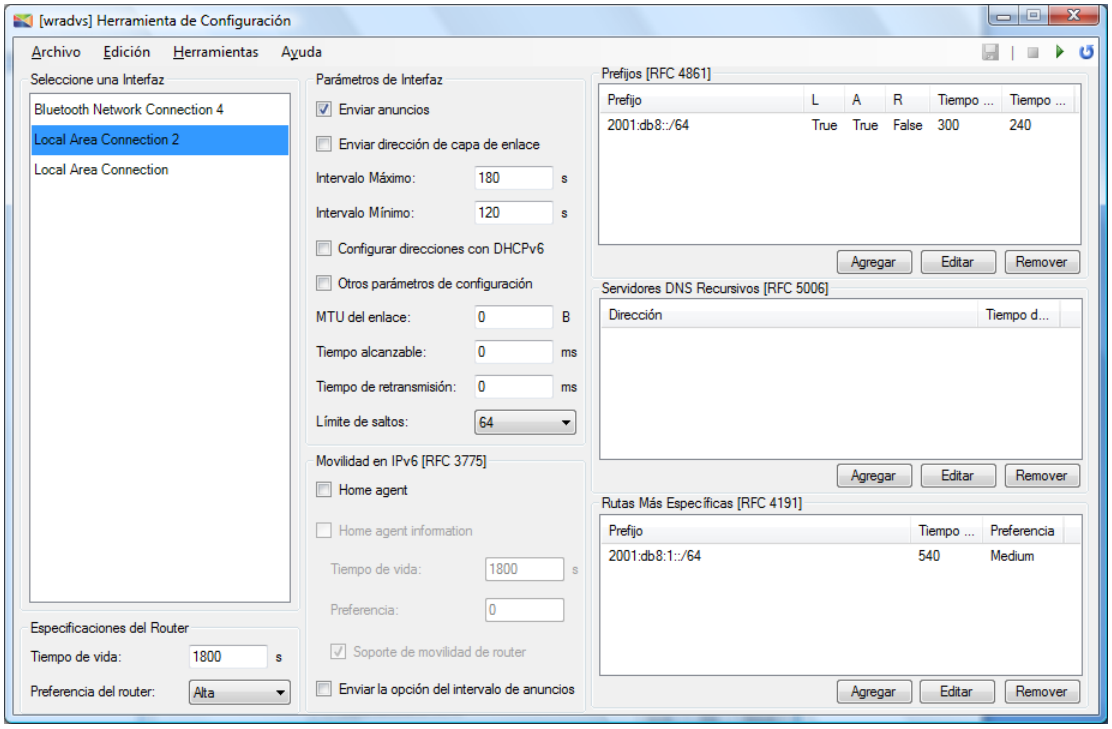

Figura 10.10: Configuración de la interfaz *Local Area Connection 2* del router D.

Finalmente se debe activar el reenvío de paquetes en ambas interfaces. Esto se realiza mediante la herramienta de configuración de direcciones IPv6 en la pestaña *Forwarding*, tal como lo muestra la Figura 10.11:

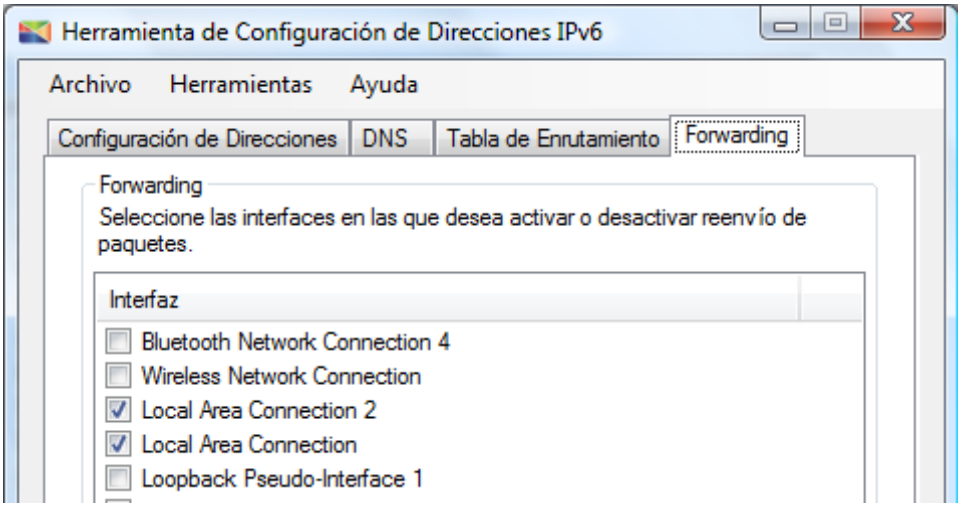

Figura 10.11: Activación de reenvío de paquetes en las interfaces del router D.

✓ **Router E**

Las dos interfaces de red del router E necesitan configurarse apropiadamente (ver Figura 10.12 y Figura 10.13):

- **Local Area Connection**

Se configuraron los siguientes parámetros de interfaz:

- **Enviar anuncios:** se selecciona la casilla para activar el envío de mensajes RA.
- **Intervalo máximo:** se le asigna 120 segundos, indicando que los mensajes RA deben ser anunciados en un intervalo de tiempo no mayor a dos minutos.
- **Intervalo mínimo:** se le asigna 60 segundos, indicando que los mensajes RA deben ser anunciados en un intervalo de tiempo no menor a un minuto.

Se configuraron los siguientes parámetros de especificaciones del router:

- **Tiempo de vida**: 1800 segundos, indicando que los hosts que reciben el mensaje RA deberán configurar al router E como router por defecto.
- **Preferencia del router**: media.

Se configuró el anuncio del siguiente prefijo:

- **Prefijo:** 2001:db8:2::/64.
- **Dentro del enlace:** sí, el prefijo se encuentra dentro del enlace.
- **Autoconfiguración sin estado:** sí, se deben generar direcciones a partir del prefijo.
- **Tiempo de vida:** 300 segundos, indicando que el prefijo será válido para determinación de rutas por 5 minutos.
- **Tiempo preferido:** 240 segundos, indicando que las direcciones generadas a partir del prefijo permanecerán en estado “preferido” por 4 minutos.

Se configuró el anuncio de las siguientes rutas más específicas:

- **Prefijo:** 2001:db8::/64.
- **Tiempo de vida:** 360 segundos, indicando que el prefijo será utilizado en determinación de rutas durante 6 minutos.
- **Preferencia:** media.
- **Prefijo:** 2001:db8:1::/64.

- **Tiempo de vida:** 360 segundos, indicando que el prefijo será utilizado en determinación de rutas durante 6 minutos.
- **Preferencia:** media.

- **Local Area Connection 2**

Se configuraron los siguientes parámetros de interfaz:

- **Enviar anuncios:** se selecciona la casilla para activar el envío de mensajes RA.
- **Intervalo máximo:** se le asigna 180 segundos, indicando que los mensajes RA deben ser anunciados en un intervalo de tiempo no mayor a tres minutos.
- **Intervalo mínimo:** se le asigna 120 segundos, indicando que los mensajes RA deben ser anunciados en un intervalo de tiempo no menor a dos minutos.

Se configuraron los siguientes parámetros de especificaciones del router:

- **Tiempo de vida**: 1800 segundos, indicando que los hosts que reciben el mensaje RA deberán configurar al router E como router por defecto.
- **Preferencia del router**: media.

Se configuró el anuncio del siguiente prefijo:

- **Prefijo:** 2001:db8::/64.
- **Dentro del enlace:** sí, el prefijo se encuentra dentro del enlace.
- **Autoconfiguración sin estado:** sí, se deben generar direcciones a partir del prefijo.
- **Tiempo de vida:** 300 segundos, indicando que el prefijo será válido para determinación de rutas por 5 minutos.
- **Tiempo preferido:** 240 segundos, indicando que las direcciones generadas a partir del prefijo permanecerán en estado “preferido” por 4 minutos.

Se configuró el anuncio de la siguiente ruta más específica:

- **Prefijo:** 2001:db8:2::/64.
- **Tiempo de vida:** 540 segundos, indicando que el prefijo será utilizado en determinación de rutas durante 9 minutos.
- **Preferencia:** media.

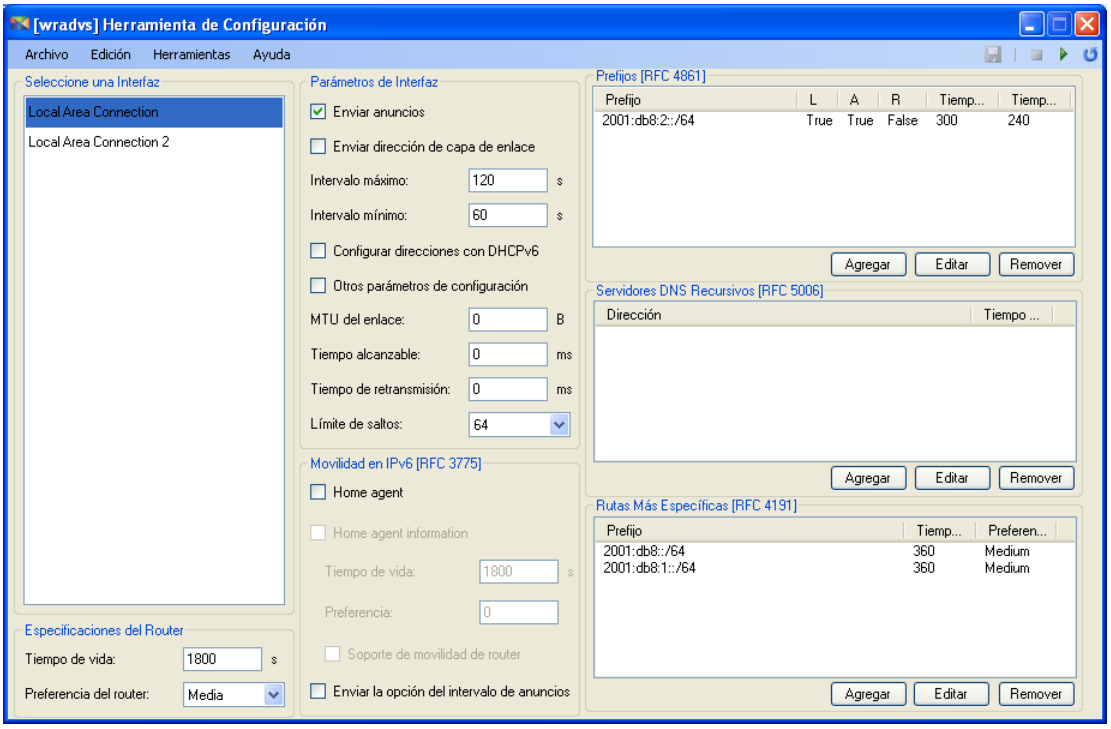

Figura 10.12: Configuración la interfaz *Local Area Connection* del router E.

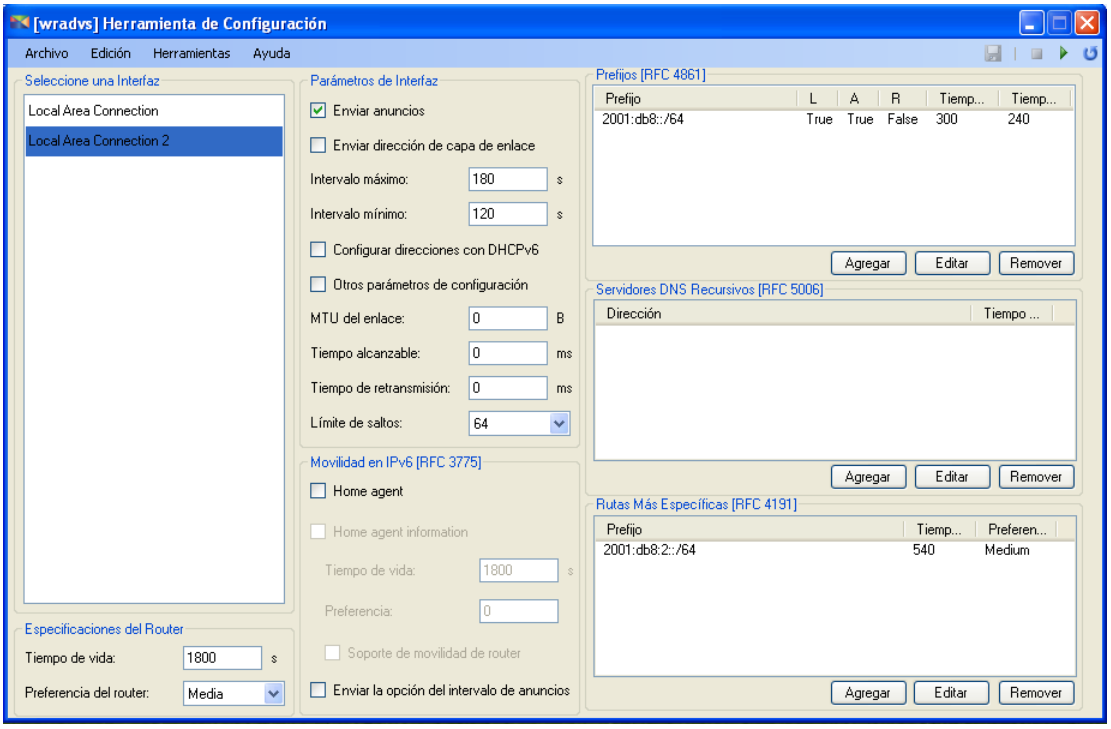

Figura 10.13: Configuración de la interfaz *Local Area Connection 2* del router E.

Finalmente se debe activar el reenvío de paquetes en ambas interfaces mediante la herramienta de configuración de direcciones IPv6, tal como lo muestra la Figura 10.11:

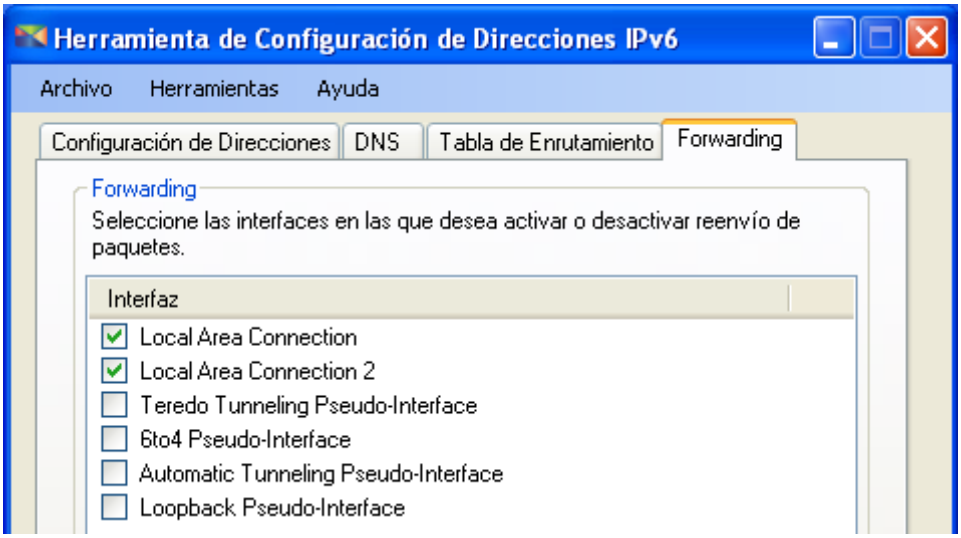

Figura 10.14: Activación de reenvío de paquetes en las interfaces del router E.

10.3 Sprint Review

10.3.1 Inicio del proceso de autoconfiguración

Finalizada la configuración de los routers, se da inicio al anuncio de los mensajes RA, mediante la opción "Herramientas -> Iniciar Servidor" de la herramienta de configuración (ver Tabla 6.8).

10.3.2 Estado final de los hosts

Luego de que los nodos hayan recibido los mensajes RA y se hayan autoconfigurados, ellos presentan las siguientes características:

✓ **Host A**

De igual manera que en el estado inicial, se utilizó el comando *"ifconfig en0 | grep inet6"* para obtener la información de la interfaz relacionada a IPv6 (ver Figura 10.15):

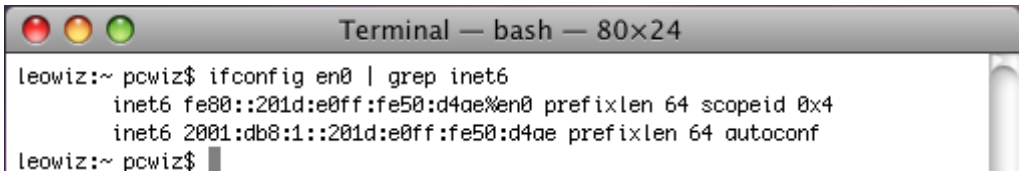

Figura 10.15: Estado final de la interfaz *en0* del host A.

Se puede comprobar que el host A recibió mensajes RA del router D (el cual anunciaba el prefijo 2001:db8:1::/64) y efectuó la autoconfiguración, obteniendo la dirección 2001:db8:1::201d:e0ff:fe50:d4ae/64 la cual fue generada a partir del prefijo especificado y la dirección de capa de enlace.

El host A no configuró rutas más específicas debido a que el sistema operativo que utiliza no posee soporte para estas opciones de los mensajes RA.

✓ **Host B**

Nuevamente al utilizar el comando *"ifconfig eth2* | *grep inet6"*, se obtuvieron las direcciones IPv6 de la interfaz *eth2* del host B (ver Figura 10.16):

```
debian:~# ifconfig eth2 | grep inet6
   inet6 addr: 2001:db8:2:0:20c:29ff:fe74:fc2b/64 Scope:Global
   inet6 addr: fe80::20c:29ff:fe74:fc2b/64 Scope:Link
debian:~# _
```

Figura 10.16: Estado final de la interfaz *eth2* host B.

Al igual que el host A, el host B recibió mensajes RA. En este caso los mensajes RA fueron enviados por el router E, el cual anunciaba el prefijo 2001:db8:2::/64, motivo por el cual el host B ahora posee la dirección IPv6 2001:db8:2::20c:29ff:fe74:fc2b/64.

El host B no realizó ningún tipo de configuración de rutas más específicas debido a que el sistema operativo que utiliza no posee soporte para estas opciones de los mensajes RA.

✓ **Host C**

En la Figura 10.17 se puede observar las direcciones IPv6 y los routers por defecto de la interfaz *Local Area Connection*:

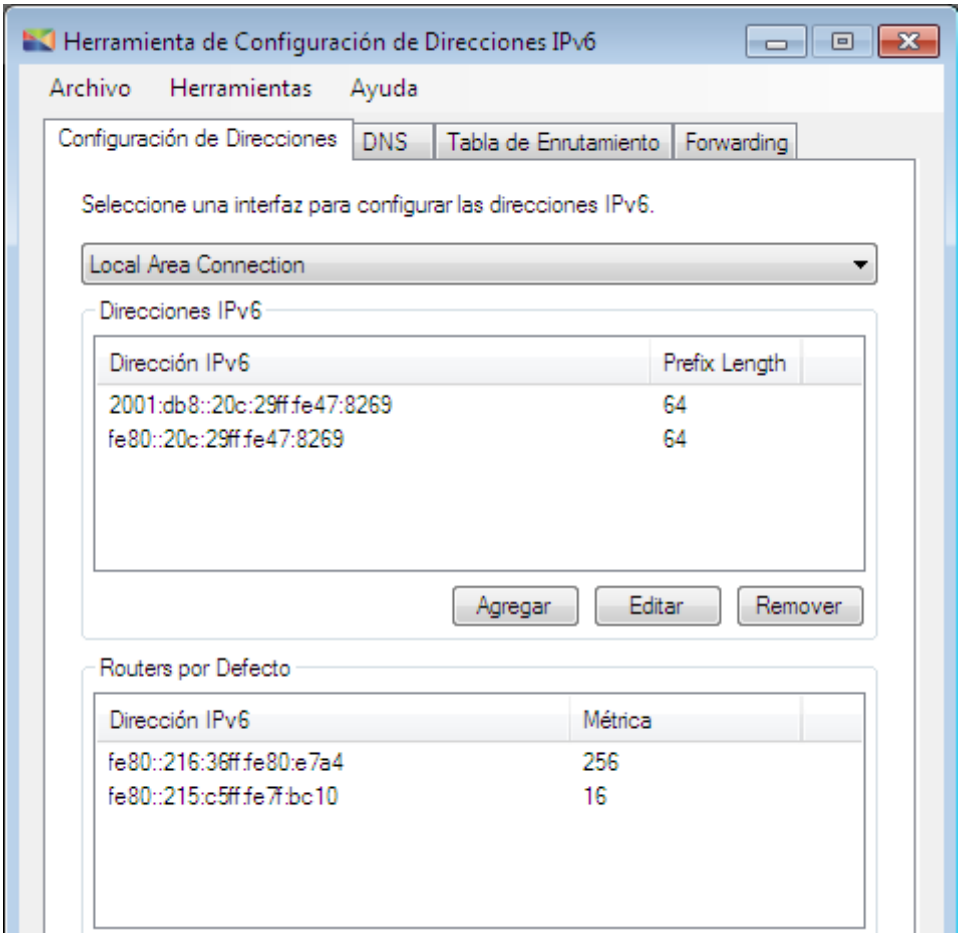

Figura 10.17: Estado final de la interfaz *Local Area Connection* del host C.

Se generó la dirección IPv6 2001:db8::20c:29ff:fe47:8269/64 (a partir del prefijo 2001:db8::/64 y la dirección de capa de enlace).

En la Figura 10.17 también se puede observar la existencia de dos routers por defecto, cuyas direcciones IPv6 corresponden a las direcciones link-local de las interfaces *Local Area Connection 2* de los routers D y E respectivamente (ver Figura 10.6 y Figura 10.8). Existe una diferencia en la métrica de ambos routers por defecto, esto se debe a que el router D hizo anuncio de los mensajes RA con la preferencia de router “Alta”, por lo tanto posee una menor métrica al router E cuyos mensajes RA anunciaban preferencia de router “Media”.

Dado que el host C utiliza Windows Seven, las rutas más específicas enviadas por los routers si fueron configuradas (ver Figura 10.18).

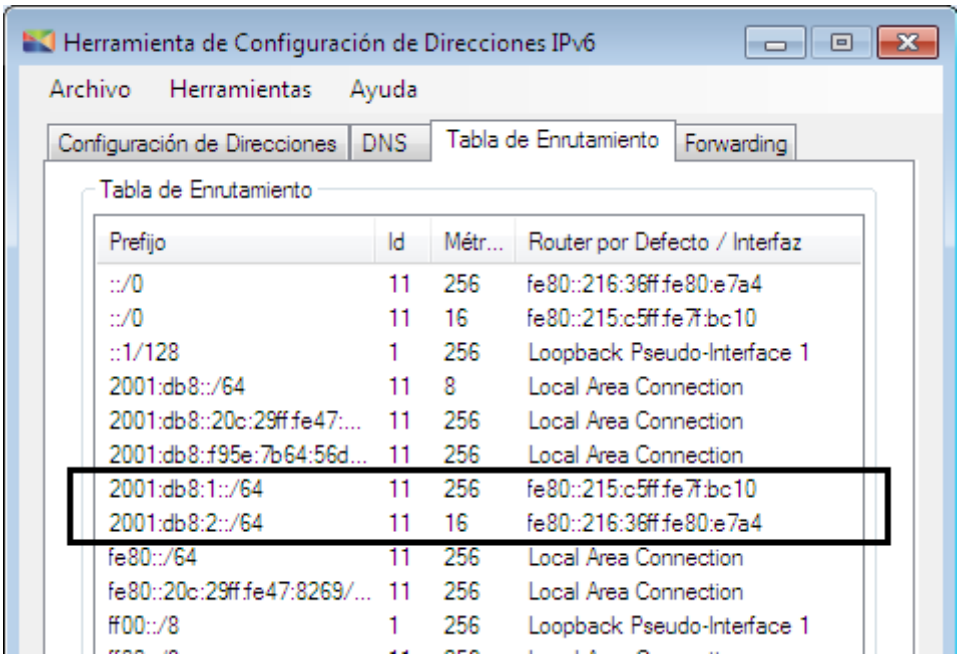

Figura 10.18: Estado final de las rutas del host C.

Las dos rutas dentro del recuadro, son las rutas más específicas anunciadas por los routers; el router D anunció la ruta 2001:db8:1::/64 y el router E anunció la ruta 2001:db8:2::/64.

10.3.3 Conectividad

Se utilizará la herramienta *ping* para comprobar la conectividad de los hosts. Previamente fue activado el reenvío de paquetes en los routers, pero para hacer posible el enrutamiento, se debe agregar direcciones IPv6 a las interfaces de los routers:

Para el router D es necesario agregar una dirección IPv6 de prefijo 2001:db8:2::/64 en la interfaz *Local Area Connection* y una dirección de prefijo 2001:db8::/64 en la interfaz *Local Area Connection 2*, así como se ve en la Figura 10.19 y la Figura 10.20.

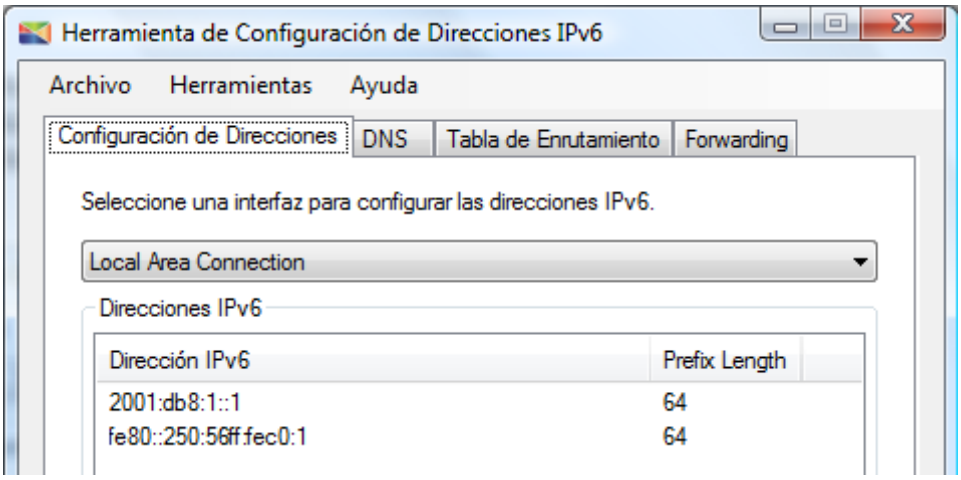

Figura 10.19: Interfaz *Local Area Connection* del router D.

Adicionalmente, se debe añadir el router E (la dirección link-local de la interfaz *Local Area Connectionc 2*) como router por defecto de la interfaz *Local Area Connection 2* del router D (ver Figura 10.20); logrando así el enrutamiento de paquetes destinados a la red 2001:db8:2::/64.

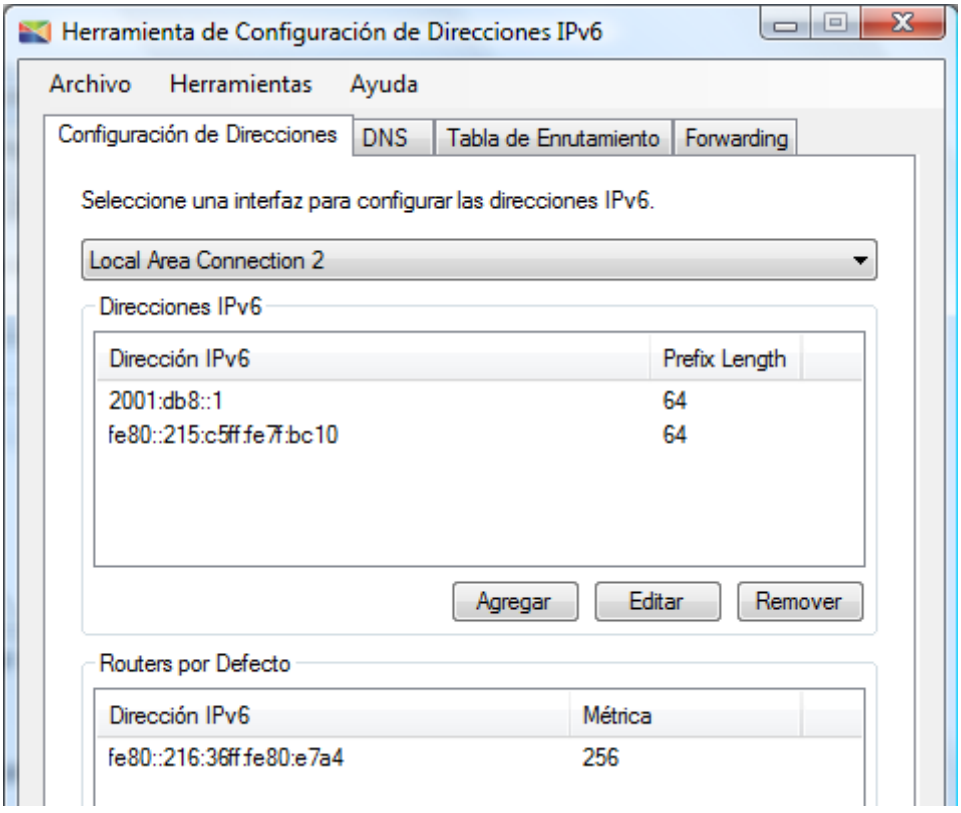

Figura 10.20: Interfaz *Local Area Connection 2 del router D.*

En el router E no es necesario añadir direcciones ni router por defecto, debido a que el sistema operativo (Windows XP) utilizado por el router no permite ignorar los mensajes RA. A causa de esto, el router E ha realizado autoconfiguración de direcciones con el prefijo 2001:db8::/64 en la interfaz *Local Area Connection (ver* Figura 10.21*)* y direcciones con el prefijo 2001:db8:2::/64 en la interfaz *Local Area Connection 2 (ver* Figura 10.22*)*.

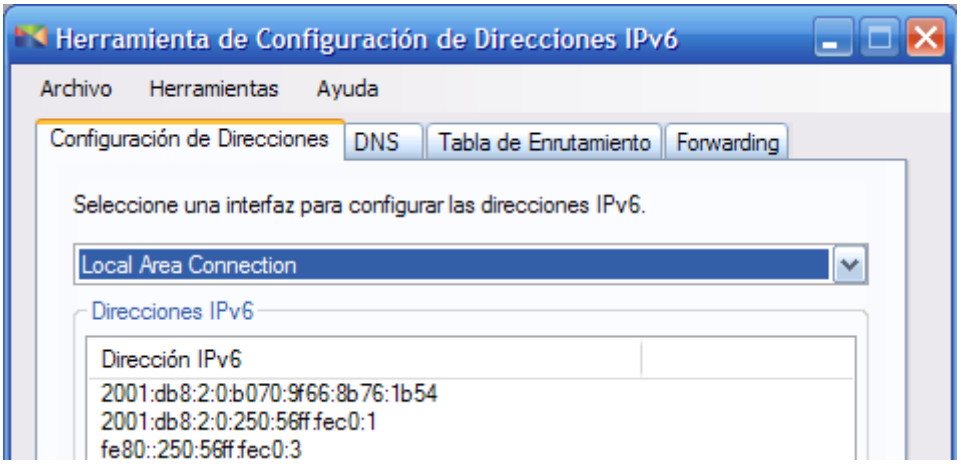

Figura 10.21: Interfaz *Local Area Connection* del router E.

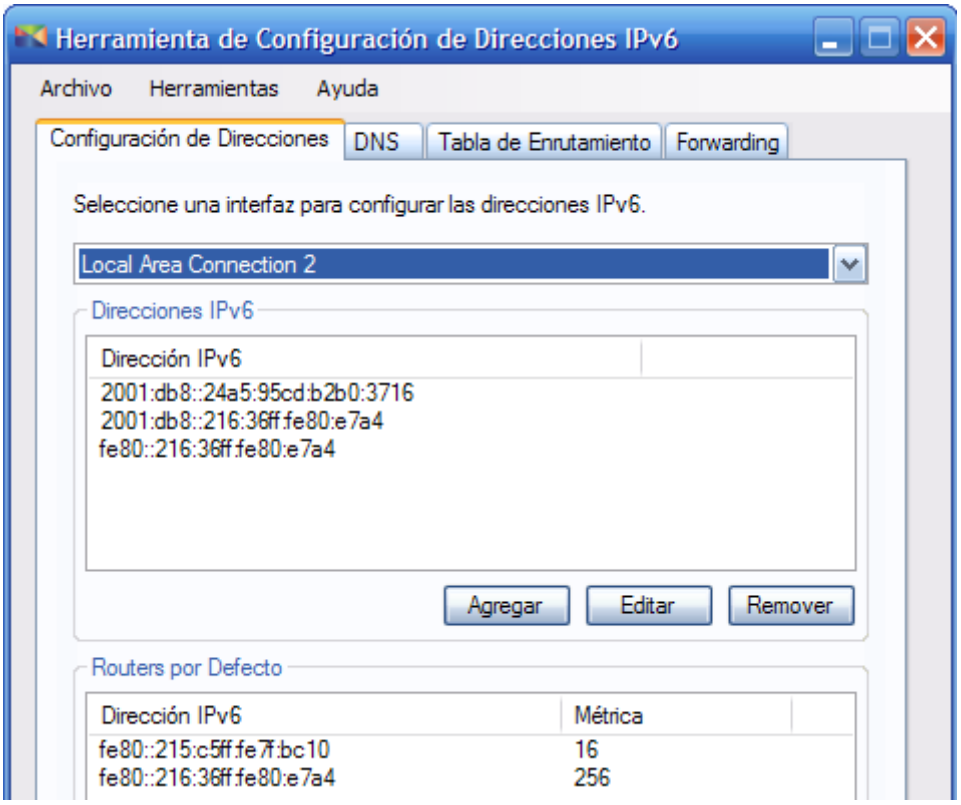

Figura 10.22: Interfaz *Local Area Connection 2* del router E.

Ya configurados todos los hosts y routers, se procede a utilizar el comando *ping* en la línea de comandos del host A (ver Figura 10.23).

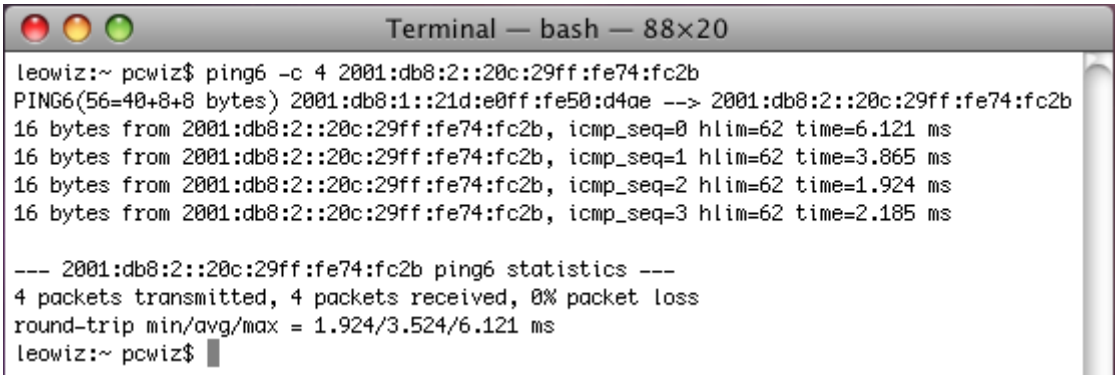

Figura 10.23: Ping entre el host A y B.

Se utilizó el comando "*ping6 –c 4 2001:db8:2::20c:29ff:fe74:fc2b*", donde "*ping6*" indica que se utilizarán direcciones IPv6, la opción "*-c 4*" indica que sólo se enviarán 4 solicitudes (*Echo Request)*, finalmente se encuentra la dirección IPv6 del host B (ver Figura 10.16).

Una vez que se realizó toda la configuración de los routers y se inició el proceso de autoconfiguración sin estado se esperó unos segundos antes de realizar la prueba de conectividad. Como resultado el *ping* fue exitoso, lo que indica que el proceso se llevó a cabo de la forma correcta permitiendo la convergencia de la red.

11. Conclusiones

El servidor de autoconfiguración sin estado Windows Router Advertisement Server (wardvs) es una aplicación de código abierto que permite la autoconfiguración IPv6 de los nodos de un enlace. Para lograrlo utiliza el envío de mensajes Router Advertisement periódicos y solicitados que soportan todos los parámetros y opciones mencionados en los RFCs.

La aplicación fue realizada utilizando Visual Studio 2008 como entorno de desarrollo y C# como lenguaje de programación. El *.Net Framework 3.5* permitió crear las interfaces gráficas, colocar la aplicación como un servicio de Windows, realizar un instalador para facilitar la herramienta a los usuarios y hacer el código más comprensible y eficiente gracias a nuevas librerías para el acceso a listas y archivos XML.

La utilización de *SharpPcap* (*wrapper* para C# de la librería *WinPcap*) facilitó el proceso de captura de los paquetes para recibir los mensajes Router Solicitation y así permitir el envío de mensajes Router Advertisement solicitados.

La realización del visor de eventos facilita el proceso de depuración en el proceso de autoconfiguración de una forma gráfica y sencilla. Además de permitir una herramienta de enseñanza gracias a la información detallada que se presenta en cada mensaje y en los tooltips que se muestran en la herramienta de configuración del servidor con información obtenida en el trabajo previo de investigación.

La herramienta de configuración de direcciones IPv6 otorga a los usuarios una interfaz gráfica para la configuración de direcciones IPv6, routers por defecto, servidores DNS, administración de la tabla de enrutamiento y activación de reenvío de paquetes.

Con esta nueva aplicación los sistemas Windows ahora poseen una completa solución para realizar la autoconfiguración sin estado, tomando en cuenta que *netsh* solamente permite la configuración de pocos parámetros y que no posee interfaz gráfica alguna.

La aplicación soporta los idiomas español e inglés y provee archivos de ayuda para que los nuevos usuarios solventen las dudas acerca del funcionamiento de la misma.

Con el fin de mantener la aplicación actualizada y mejorarla en la medida de lo posible el servidor de autoconfiguración sin estado Windows Router

Advertisement Server ha sido publicado bajo licencia *GNU GPL* en el sitio web *SourceForge.net* que permite la distribución de proyectos de software libre y código abierto. El enlace para la descarga de la aplicación es: *http://wradvs.sourceforge.net.*

El objetivo principal del Trabajo Especial de Grado fue realizar el servidor de autoconfiguración sin estado, la herramienta gráfica para configurarlo y el visor de eventos para depurar posibles fallas de configuración, en este sentido la aplicación cumple con todos los objetivos planteados. Sin embargo, la herramienta de configuración de direcciones IPv6 que se agregó al proyecto utiliza *netsh* como motor de ejecución para permitir que la aplicación funcione por completo tanto en Windows XP como en Vista.

Tomando en cuenta esta información se proponen algunos trabajos que podrían complementar la aplicación:

- ✓ Realizar una librería que unifique el proceso de configuración de direcciones IPv6 en los diferentes sistemas Windows, sin la utilización de *netsh* como motor. Como base para la realización de dicho proyecto se recomienda el uso de *IP Helper*[6], una librería para la configuración de IP e IPv6. Por los momentos, la librería proporciona funciones que no han sido implementadas, sin embargo, futuros proyectos indican que pronto estará a disposición de los desarrolladores una versión más completa en lenguaje C++.

- ✓ En general, se pueden añadir nuevos idiomas a la aplicación con facilidad realizando nuevos archivos de idioma. Por lo que se puede hacer que la aplicación soporte cualquier idioma que se desee.

- ✓ Realización de un cliente para Windows XP que analice los mensajes RA recibidos y configure los servidores DNS recursivos y las rutas más específicas, dado que Windows XP no procesa las opciones para realizar dichas configuraciones.

[6] http://msdn.microsoft.com/en-us/library/aa366073(VS.85).aspx

Referencias Bibliográficas

[1] S. Gunderson. Google: Global IPv6 Statistics. RIPE 57. October, 2008.

[2] S. Deering, R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460. Diciembre, 1998.

[3] S. Hagen. IPv6 Essentials, 2nd Edition. O'Reilly. Mayo, 2006.

[4] J. Davies. Understanding IPv6, 2nd Edition. Editorial: Microsoft Press, Febrero, 2008.

[5] S. Kent. IP Encapsulating Security Payload (ESP). RFC 4303. Diciembre, 2005.

[6] R. Hinden, S. Deering. IP Version 6 Addressing Architecture. RFC 4291. Febrero, 2006.

[7] B. Carpenter, K. Moore. Connection of IPv6 Domains via IPv4 Clouds. RFC 3056. Febrero, 2001.

[8] R. Hinden, B. Haberman. Unique Local IPv6 Unicast Addresses. RFC 4193. Octubre, 2005.

[9] B. Stockebrand. IPv6 in Practice. Springer. Febrero, 2007.

[10] A. Conta, A. Deering, M. Gupta. Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. RFC 4443. Marzo, 2006.

[11] R. Bonica, D. Gan, D. Tappan, C. Pignataro. Extended ICMP to Support Multi-Part Messages. RFC 4884. Abril, 2007.

[12] T. Narten, E. Nordmark, W. Simpson, H. Soliman. Neighbor Discovery for IP Version 6 (IPv6). RFC 4861. Septiembre, 2007.

[13] R. Draves, D. Thaler. Default Router Preferences and More-Specific Routes. RFC 4191. Noviembre, 2005.

[14] D. Johnson, C. Perkins, J. Arkko. Mobility Support in IPv6. RFC 3775. Junio, 2004.

[15] J. Jeong, S. Park, L. Beloeil, S. Madanapalli. IPv6 Router Advertisement Option for DNS Configuration. RFC 5006. Septiembre, 2007.

[16] V. Devarapalli, R. Wakikawa, A. Petrescu, P. Thubert. Nework Mobility (NEMO) Basic Support Protocol. RFC 3963. Enero, 2005.

[17] S. Thomson, T. Narten, T. Jinmei. IPv6 Stateless Address Autoconfiguration. RFC 4862. Septiembre, 2007.

[18] I. van Beijnum. Running IPv6. Apress. Noviembre, 2005.

[19] R. Ed, J. Bound, B. Volz, T. Lemon, C. Perkins, M. Carney. Dynamic Host Configuration Protocol for IPv6 (DHCPv6). RFC 3315. Julio, 2003.

[20] O. Troan, R. Droms. IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6. RFC 3633. Diciembre, 2003.

[21] K. Schwaber. Agile Project Management with Scrum. Microsoft Press. Marzo, 2004.