



*Benemérita Universidad
Autónoma de Puebla.
Facultad de Ciencias de la Computación.*

*Técnicas de comunicación IPv6
en redes IPv4.*

Tesis

que para obtener el grado de:

*Maestro en ciencias de la
Computación.*

Presenta:

Martín Oswaldo López Carrera.

Asesor

M. C. Jesús García Fernández

Puebla, Pue.

Diciembre de 2003.

Agradecimientos:

Con cariño y respeto dedico esta Tesis a:

Mis padres Manuel y Lupita.

Por su incondicional apoyo en todo momento y en particular en la culminación de esta importante etapa de mi vida.

Porque confiaron en mí...

Gracias...

Mis hermanos Susana Walter e Ivonne.

Por todos los momentos que hemos compartido...

Mi Asesor M. C. Jesús García Fernández.

Por su completo apoyo y paciencia en la elaboración de este documento.

M. C. David Eduardo Pinto Avendaño.

Porque sin su incondicional ayuda no hubiera logrado tantas cosas buenas que tengo...

Sinceramente Gracias.

Resumen

En este documento se presenta un estudio del protocolo IPv6 (Internet Protocol version 6), describiendo las principales características, mejoras, aplicaciones, formato de direcciones, requerimientos, etc. También se describen los mecanismos de transición entre formatos y estrategias de transición para conectar dos host mediante el protocolo IPv6 utilizando una red IPv4 (Internet Protocol version 4). Además se presenta información importante de túneles (*tunneling*), fragmentación y los mecanismos comunes entre éstos.

Otro aspecto importante que aquí se presenta, es el ruteo en IPv4 e IPv6, resaltando las características principales, además de describir aspectos relevantes del ruteo en el *tunneling*, prioridades, fragmentación, manejo de errores, entre otros.

El enfoque de este documento está centrado en la técnica de *tunneling* para realizar un enlace entre dos redes IPv6 que cruzan por una red IPv4. Aquí se describen los métodos existentes y se realiza una caracterización exhaustiva de esta técnica resaltando sus principales cualidades así como sus desventajas sobre los métodos existentes, también se presenta una aplicación que realiza *tunneling*.

La parte fundamental del documento describe ampliamente la implementación de un mecanismo de *tunneling* para enlazar dos redes IPv6 que deben atravesar por una red IPv4. En este punto se realiza un análisis completo presentando detalladamente el procedimiento realizado, desde las herramientas indispensables, el diseño básico de la red, características de configuración de los nodos, y del ruteador, pruebas preliminares mínimas, pruebas realizadas, problemas encontrados y sus soluciones, etc.

Este método también puede ser utilizado como parte de una solución que permita a un host IPv6, el cual no tiene una dirección IPv4 asignada permanentemente, comunicarse con un host con dirección IPv4 exclusivamente.

Además, se presenta una descripción detallada de las razones de la elección de las herramientas, del diseño de la subred, del tipo de direcciones, del tipo de túnel utilizado, y algunos otros aspectos importantes.

Finalmente, en los apéndices, se presentan descripciones detalladas de algunas características importantes del protocolo IPv6 que vale la pena profundizar, tales como las extensiones de encabezado, características detalladas de los túneles, aspectos de configuración y pseudo código de los programas utilizados.

ÍNDICE

Prefacio	vi
CAPÍTULO 1 Protocolo IPv6.	1
1.1 La crisis del protocolo IPv4.	1
1.2 Estructura de direcciones IPv6.	3
1.2.1 Formato del encabezado IPv6.	4
1.2.2 La filosofía de las direcciones IPv6.	6
1.3 Especificaciones de IPv6.	11
1.4 Tamaño del paquete.	12
1.5 Requerimientos de direcciones para los nodos.	12
1.6 Compatibilidad de direcciones.	13
1.7 El futuro de IPv6.	14
CAPÍTULO 2 Mecanismos de migración.	17
2.1 Perspectiva de transición.	17
2.2 Tunneling.	18
2.3 Tipos de túneles.	23
2.3.1 Mecanismos comunes de los túneles.	26
2.4 Estrategias de diseño.	27
2.5 El futuro del Tunneling IPv6.	28
CAPÍTULO 3 Ruteo y Fragmentación.	32
3.1 Mecanismos de Ruteo.	32
3.2 Ruteo en IPv4 e IPv6	32
3.3 Ruteo.	33
3.3.1 Procesamiento de solicitudes y respuestas.	37
3.4 Ruteo en los túneles.	39
3.5 Prioridades y Clase de Servicio.	40
3.6 Fragmentación.	41
3.7 Manejo de errores.	43
3.7.1 Mensajes ICMPv6.	44
CAPÍTULO 4 Implementación.	48
4.1 Introducción.	48
4.2 Configuración e instalación.	49
4.2.1 Especificaciones en el diseño de la subred.	52

4.3	Pruebas preliminares.	53
4.4	Implementación.	54
4.5	Resultados obtenidos.	60
4.6	Aspectos relevantes.	63
4.7	QoS y Consideraciones de seguridad.	65
	Conclusiones.	67
Apéndice A.	<i>Estructura de direcciones IPv6.</i>	71
A.1	Direcciones IPv6.	71
A2.	Las extensiones del encabezado.	73
Apéndice B.	<i>Fragmentación y Tunneling.</i>	83
B.1	Fragmentación.	83
B.2	Tunneling.	85
Apéndice C.	<i>Pseudo códigos.</i>	87
Bibliografía.		89

Prefacio

El modelo TCP (Transmission Control Protocol) utilizando el protocolo IPv4 es uno de los modelos más ampliamente utilizados en el enlace entre redes. Es caracterizado por proveer conectividad global a todo aquel que lo requiera, en particular en el uso de Internet. Considerando a Internet como un medio de interconexión de redes físicas, éste presenta tres obstáculos principales: conexiones dinámicas, pérdida de datos y caminos reducidos [WaS01]. Cuando un host envía un mensaje a otro, el mensaje viaja a través de redes, atravesando *ruteadores* y *gateways*, cada mensaje enviado puede utilizar un camino distinto. Los segmentos de red a menudo aparecen y desaparecen cuando los servidores inician o se apagan.

La potencia de Internet radica en su capacidad para adaptarse a estos cambios y encaminar la información de forma consecuente. Cuando el destino obtiene el mensaje, verifica la integridad de los datos. Ya que los datos pueden viajar por caminos de comunicación poco óptimos, éstos pueden desechar o corromper los bits del mensaje. Aún más, si se trata de un mensaje demasiado grande para atravesar una red, lo cual comúnmente se debe a las diferentes tecnologías de interconexión, éste tendrá que fragmentarse.

La fragmentación de un paquete es un proceso que debe realizar un ruteador IPv4 (en IPv6 es un proceso distinto), es decir, cuando un paquete es demasiado grande para atravesar dos redes unidas por al menos un ruteador intermedio, el ruteador que recibe el mensaje, compara el tamaño del mensaje con el MTU (Maximum Transfer Unit) que puede transmitir dicho ruteador. Si el mensaje supera el tamaño del MTU, el paquete tendrá que fragmentarse, teniendo en consideración que éste mismo debe reensamblarse al llegar al destino. En el caso de estar en la misma subred, el protocolo TCP computará el tamaño máximo de segmento, de tal forma que los datagramas resultantes correspondan con la MTU de la red.

Aunado a las limitaciones presentadas anteriormente, se presentó el problema de espacio de direcciones en IPv4, al cual se le aplicó un remedio *temporal*. La solución provisional fue de asignar múltiples direcciones de clase C en lugar de direcciones de clase B. Extendiendo así un poco más el espacio de direcciones [ReY+93a] [ReY+93b]. Esta solución tiene la desventaja de expandir alarmantemente el tamaño de las tablas de ruteo en los ruteadores.

A pesar de las limitaciones mencionadas anteriormente, IPv4 tiene sus ventajas, entre ellas se encuentra uno de los aspectos más importantes de Internet, el cual es proporcionar una trayectoria global en la capa de Internet, y por lo tanto, es posible en dicha capa mantener un punto común entre todos los nodos de Internet [MiM99]. En efecto, su principal objetivo es proveer un servicio de *enlace* a todo aquel que lo requiera.

Cuando las proyecciones indicaban que el espacio de direcciones crecía a un ritmo mayor de lo calculado, se buscaron maneras de resolver las limitaciones del espacio de direcciones, mientras al mismo tiempo se proveía de funcionalidad adicional al protocolo

existente (IPv4), esto es, se modificaba la forma en que se manejaba la "máscara de red", con el fin de proporcionar un espacio extra de direcciones, para satisfacer un poco mas la solicitud continua de direcciones. Ello motivó a la IETF (*Internet Engineering Task Force*) a la creación de un grupo que pudiera caracterizar y desarrollar al sucesor de IPv4.

En 1993 la IETF formó el grupo llamado "IPng Area", y en diciembre de 1993 fue distribuido un artículo titulado *IP: Next Generation (IPng) White Paper Solicitation* [BrS+93], el cual invitó a todo aquel interesado en participar para especificar lo requerimientos que tiene que satisfacer IPng. De esta publicación, se obtuvieron 24 recomendaciones. Además, se publicó en diciembre de 1994 el artículo titulado *Technical Criteria for Choosing IP The Next Generation (IPng)* [PaC+94] el cual define un conjunto de criterios a ser utilizados en el Proceso de evaluación de IPng.

En enero de 1995 se publicó el artículo titulado *The Recommendation for the IP Next Generation Protocol*, el cual analizó la recomendaciones hechas al protocolo, y especificó 17 recomendaciones, las cuales incluyen entre otras, un encabezado simplificado y una estructura de direcciones jerárquica que permita asignación rigurosa de ruta y así mismo se pueda extenderse para satisfacer las necesidades de Internet en un futuro. El protocolo también debe incluir autenticación y encriptación a nivel de paquetes, junto con una *configuración automática*. [BrS+95]. Este artículo se resume en tres evaluaciones de las propuestas IPng: CATNIP, SIP y TUBA.

De las tres propuestas anteriores, se sacaron las mejores capacidades para obtener IPng. Posteriormente, se publicó el artículo *Internet Protocol, Version 6 (IPv6) Specification* [DeS+95], ahora obsoleto y substituido por una versión posterior [DeS+98]. Este fue el nacimiento de IPv6, y de ahí se desprendieron múltiples artículos proporcionando mejoras al protocolo y a cada uno de los campos de su encabezado.

Esencialmente el protocolo IPv4 es un subconjunto del protocolo IPv6, así que este último hereda todas las características buenas de su predecesor y desecha las anticuadas. Así, es posible pensar en IPv6 como "un protocolo para enlazar a todos". La principal diferencia entre IPv4 e IPv6 es la aparición de direcciones *anycast* en IPv6 y la desaparición de direcciones *broadcast* en IPv4 reemplazadas por direcciones *multicast* en IPv6.

Esto no indica que exista solo un protocolo de Internet, ya que se trata de un Internet multiprotocolos.

IPv6 ofrece servicios semejantes a IPv4 pero mejorando los bien conocidos problemas de escalamiento en la arquitectura de Internet a mas nodos finales y aún mejorar el incremento de ancho de banda.

Básicamente IPv6, provee:

- Auto configuración.
- Simplicidad de arquitectura.

- Encabezado simplificado y flexible.
- Extensión de encabezado simplificado.
- Estructura de direcciones que permita alcance riguroso de una ruta.
- Espacio de direcciones suficientemente grande para satisfacer las necesidades futuras a largo plazo.
- Autenticación y privacidad.
- Encriptación.
- Posibilidad de incrementar nuevas opciones en el futuro, en un proceso de mejora constante.
- Capacidades de Calidad de Servicio son agregadas para etiquetar los paquetes pertenecientes a un flujo de paquetes en particular, a los cuales la fuente solicita un tratamiento especial, tales como video, etc.

De manera similar a IPv4, en IPv6 también debe realizarse fragmentación y reensamble. Aunque a diferencia de IPv4, en IPv6 la fragmentación no se realiza en un ruteador intermedio, sino en el nodo fuente, y el reensamble se realiza en el nodo destino, con lo cual debe preparar al destino final para realizar el reensamble. El realizar la fragmentación en el nodo fuente, requiere que exista una técnica para conocer la MTU a lo largo de la trayectoria hasta el destino, evitando así fragmentaciones adicionales en algún ruteador intermedio.

Esta característica proporciona flexibilidad en el manejo de datagramas, puesto que al ruteador se le quita carga de CPU al no realizar la fragmentación, con lo cual se logra mejorar el flujo de transporte.

También es necesario tomar en consideración el ruteo, ésto es, la selección de la trayectoria a seguir por el datagrama desde la dirección fuente hasta la dirección destino. En IPv4 se manejan dos tipos de transmisión, *directa* e *indirecta*: en la primera se envía el datagrama dentro de la misma subred, y en la segunda se debe utilizar algún ruteador intermedio, esto es, el destino no se encuentra sobre la misma subred, obligándose a entregar el datagrama a algún ruteador.

En IPv6, los mecanismos de ruteo, deben involucrar el ruteo en la *capa dual IP*, es decir, en la capa donde coexisten ambos protocolos IPv4 e IPv6. Aún más, es necesario realizar el cálculo de la ruta en esta capa. En este esquema de transición básico de *capa dual IP*, los ruteadores deben soportar independientemente el ruteo IPv4 e IPv6. Además, es necesario manejar el soporte para DNS (Domain Name System). Éstas características de transmisión de paquetes IPv6, implican el aprendizaje basado en rutas a través de ejecutar los protocolos específicos de ruteo IPv6.

CAPÍTULO 1 Protocolo IPv6.

1.1 La crisis del protocolo IPv4.

Analizando brevemente algunas características del predecesor de IPv6 (IPv4), se sabe que la estructura de direcciones IPv4, divide a la dirección IP de 32 bits en dos secciones: una sección de identificación de red y otra sección de identificador de host [PoJ81]. El tamaño de cada sección depende del número de bits que son asignados a esta. El formato del identificador es diferente para cada asignación de bits de red y host, y es identificado por los tres primeros bits.

Existen cinco clases de redes: las redes clase A, B, C, D y E. en la figura 1.1 se ilustra la estructura de éstas. Las direcciones IP están divididas en dos campos, que identifican la red y el host.

En las redes clase A se pueden formar 126 redes con 16'777,214 host cada una y son diseñadas para redes grandes, en las redes clase B se tienen 16,382 redes con 65,534 host cada una, y en las redes clase C se tienen 2'097,152 redes con 254 host cada una y son generalmente utilizadas para redes pequeñas, tales como las redes de área local “**Local Area Network (LAN)**”.

Este esquema de direccionamiento funcionó adecuadamente hasta finales de los 80's, fue entonces cuando observaron que las proyecciones indicaban que el espacio de direcciones estaba creciendo a un ritmo mayor de lo calculado, puesto que este esquema había sido diseñado sin prever el explosivo crecimiento de Internet.

Otro problema que surgió, fue el aumento en el tamaño de las tablas de ruteo en los ruteadores. Puesto que existía una gran demanda de redes LAN, se motivó a la comunidad de Internet a revisar la estructura de direcciones.

Debido a la escasez de direcciones IPv4 clase B, la IETF tuvo que encontrar una solución al problema. Surgieron dos propuestas, la solución a largo plazo era la de rediseñar el protocolo IP, lo cual no era muy simple y se adoptó una solución a corto plazo.

Así surgió la idea de agregar un campo adicional que permitiera identificar a una subred dentro de un identificador de red asignado. Ello repercutiría en reducir el campo identificador de host. Así mismo para identificar entre varias subredes se utilizó una máscara de dirección de red, además, para realizar una decisión de ruteo, el dispositivo deberá usar la máscara de red.

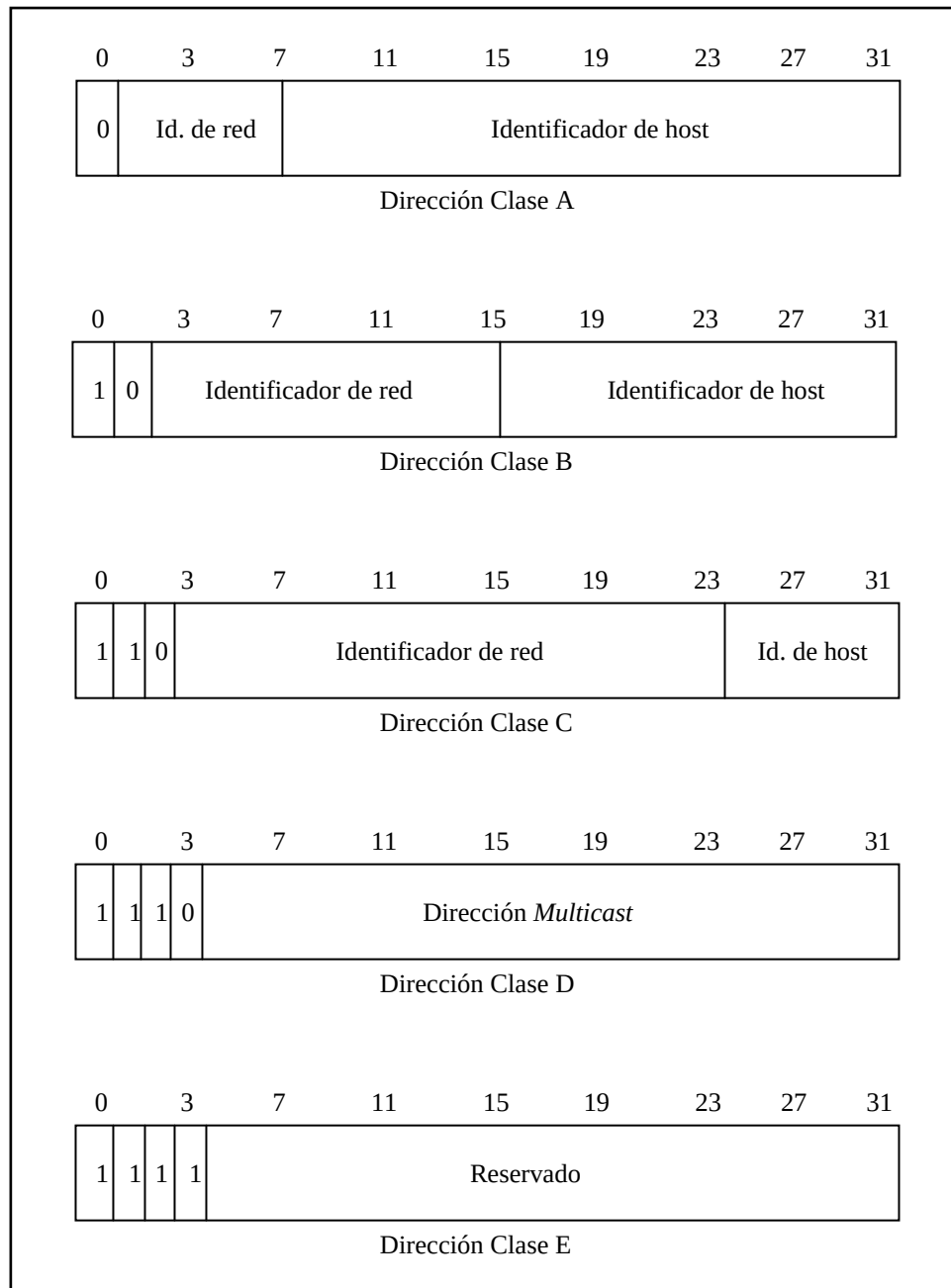


Figura 1.1. Estructura de las direcciones IPv4.

En caso de no conocer la máscara de red, se deberá utilizar el protocolo ICMP (Internet Control Message Protocol) para descubrirla, lo cual requiere esperar una respuesta del *gateway* vecino al enviarle un mensaje de difusión.

La solución temporal mas acertada, fue la de introducir el concepto de *Classless Interdomain Routing (CIDR)* [FuV+93] .

El objetivo de CIDR es el de asignar bloques contiguos de direcciones clase C a organizaciones y proveedores de servicios que necesitaran mas de 254 host (una red clase C) pero menos de 65,534 host (una red clase B). Las clases contiguas de direcciones deberán asignarse de acuerdo a la tabla 1.1.

Direcciones Clase C contiguas	Máximo Numero de host (teórico)
1	256
2	512
4	1,024
8	2,048
16	4,096

Tabla 1.1. Localización de direcciones con CIDR

Existen dos componentes básicos de este plan de ruteo y direccionamiento:

1. Distribuir la localización del espacio de direcciones de Internet.
2. Proveer un mecanismo para la incorporación de información de ruteo.

Uno de los principales objetivos de este plan de direccionamiento, es localizar el espacio de direcciones de Internet de tal manera que permita el correcto manejo de la información de ruteo.

El direccionamiento en una gran red debe ser jerárquico, con una profundidad tan grande como la amplitud de la red. Comparando el direccionamiento en IPv4, este mecanismo jerárquico solamente cuenta con tres niveles: red, subred y host. Y debido a que los ruteadores deben tener una entrada en sus tablas de ruteo para todas las redes IP existentes, se resolvió temporalmente este problema utilizando el *CIDR*.

Como característica predeterminada, IPv4 no permite indicar de manera práctica el tipo de datos transportados, y por tanto, la gestión de la urgencia o el nivel de servicio deseado deberá realizarse estableciendo expresamente características *no default* (TOS, Type Of Service, en IPv4). Esto es necesario particularmente en aplicaciones de tiempo real (como video), y en general para los tipos de servicios que así lo requieran.

1.2 Estructura de direcciones IPv6.

La estructura de direcciones IPv6 encuentra sus raíces en el *CIDR* y difiere de la estructura de direcciones IPv4. Esta estructura incluye un identificador de sitio, un identificador de host y múltiples prefijos de dirección.

Cada uno de los prefijos de dirección mencionados, puede tener múltiples estructuras similares a un identificador de sitio y un identificador de host.

Esta estructura de direcciones IPv6 está organizada utilizando el formato de prefijos, de manera similar a un número telefónico, en donde se tienen códigos de área y de país, que lógicamente dividen este tipo de prefijos en la forma de un árbol, de tal forma que una ruta de una red a otra puede ser fácilmente localizada. En la tabla 1.2 se muestra esta estructura.

Asignación.	Prefijo (bin).	Inicio del rango de direcciones.	Longitud de la máscara (bits).	Fracción del espacio de Direcciones.
Reservado	0000 0000	0 :: /8	8	1/256
Reservado para NSAP	0000 001	200 :: /7	7	1/128
Reservado para IPX	0000 010	400 :: /7	7	1/128
Dirección <i>unicast</i> global agregable	001	2000 :: /3	3	1/8
Dirección <i>unicast</i> de enlace local	1111 1110 10	FE80 :: /10	10	1/1024
Dirección <i>unicast</i> de enlace de sitio	1111 1111 11	FEC0 :: /10	10	1/1024
Dirección <i>multicast</i>	1111 1111	FF00 :: /8	8	1/256
Asignación total				15 %

Tabla 1.2. Estructura de direccionamiento de IPv6.

Los únicos prefijos que deberán ser predefinidos en una implementación son los siguientes:

- Direcciones *No especificadas*.
- Dirección *loopback*.
- Prefijo *multicast*.
- Prefijos de uso local (el *enlace local* y el *sitio local*).
- Direcciones *multicast* predefinidas.
- Prefijos con compatibilidad IPv4.

1.2.1 Formato del encabezado IPv6.

El formato del encabezado IPv6 se ilustra en la figura 1.2, y cada campo del encabezado IPv6 se explica brevemente a continuación:

1	4	8	12	16	24	32
Version	Traffic Class		Flow Label			
Payload Length				Next Header		Hop Limit
Source Address						
Destination Address						

Figura 1.2. Formato del encabezado IPv6.

- El campo **Version** de 4 bits, es el número de versión del protocolo de Internet.
- El campo **Traffic Class** de 8 bits, es utilizado por el nodo fuente y por los ruteadores para identificar y distinguir entre diferentes clases de prioridades de paquetes IPv6. Este reemplaza las funciones provistas por el campo **Type of Service** en IPv4. esta función es comúnmente referida como **Differentiated Services** Se aplican tres requerimientos generales a este campo:
 - Para paquetes que son originados dentro de un nodo por cualquier protocolo de capa superior, este protocolo deberá especificar el valor de los bits del campo Traffic Class.
 - Los nodos que soportan una función particular que utilizan los bits Traffic Class pueden cambiar los valores de estos bits en el paquete que ellos originan, transmiten o reciben. Si un nodo no soporta esta función, este no deberá cambiar los bit's de este campo.
 - Cualquier protocolo de capa superior, deberá tener en cuenta que el valor del campo Traffic Class es probablemente modificado en el transcurso del viaje, por tanto se debe considerar que el valor de este campo puede no ser el mismo en el origen que en el destino.
- El campo **Flow Label** de 20 bits, es utilizado por la fuente para etiquetar secuencias de paquetes para los cuales solicita manejo especial por los ruteadores IPv6, tales como QoS (Quality of Service) *no default* o servicio *en tiempo real*. Este campo es todavía experimental y esta sujeto a cambios a medida que los requerimientos para soporte de flujos se hagan más concisos.
- El campo **Payload Length** de 16 bits, proporciona el tamaño del resto del paquete que sigue al encabezado IPv6 en bytes. Es importante notar que cualquier extensión del encabezado (campo **Next Header**) dentro del paquete IPv6, será considerado parte de la carga, es decir, incluido en el conteo del **Payload Length**.
- El campo **Next Header** de 8 bits, identifica el tipo de encabezado que sigue inmediatamente al encabezado IPv6, utiliza los mismos valores que el campo **Protocol** en IPv4.

- El campo **Hop Limit** de 8 bits, indica el número de saltos que el paquete debe realizar y es decrementado en uno por cada nodo que transmite el paquete. Si este campo es decrementado a cero, el paquete es descartado.
- El campo **Source Address** de 128 bits, es la dirección del emisor de paquetes.
- El campo **Destination Address** de 128 bits, es la dirección del receptor del paquete, aunque posiblemente no sea el último nodo destino si es que esta presente en el campo **Next Header** la dirección de un ruteador.

1.2.2 La filosofía de las direcciones IPv6.

La representación de una dirección IPv6 tiene el siguiente formato:

X : X : X : X : X : X : X : X

donde X es un número hexadecimal de 16 bits. Esta notación puede ser simplificada cuando X tiene el valor de cero, quitando las X's = 0 y substituyéndolas por "::". Esta substitución deberá realizarse una sola vez. También es común añadir a esta dirección su prefijo:

Dirección_IPv6 / longitud_del_prefijo

donde la dirección es expresada como anteriormente se indico y la longitud del prefijo es un valor decimal que especifica el número de bits mas a la izquierda que se tomaran de la dirección en su forma expandida. La dirección "**loopback**" es asignada fuera del formato del espacio del prefijo. Además los bits de más a la izquierda, llamados el formato del prefijo, definen el tipo específico de dirección IPv6.

Se definen tres tipos diferentes de direcciones IPv6 [HiR+98a], las cuales se describen a continuación:

- **Direcciones Unicast.** Se tiene un identificador a una sola interfaz. Así un paquete enviado a una dirección *unicast* es enviado a la interfaz identificada por esa dirección.
- **Direcciones Anycast.** Se tiene un identificador para un conjunto de interfaces y típicamente pertenecen a diferentes nodos. Un paquete enviado a una dirección *Anycast* es enviado a una de las interfaces identificadas por esa dirección, normalmente es la mas cercana de acuerdo a la métrica de distancia del protocolo de ruteo.
- **Dirección Multicast.** Se tiene un identificador para un conjunto de interfaces y típicamente pertenecen a diferentes nodos. Un paquete enviado a una dirección *Anycast* es enviado a todas las interfaces identificadas por esa dirección.

A diferencia de IPv4, en IPv6 no existe “**broadcast**”, ya que esta fue reemplazada por “**multicast**”. También es importante notar que las direcciones de todos los tipos son asignadas a “**interfaces**” no a “**nodos**”, además, un nodo puede tener múltiples interfaces (multihomed), y por lo tanto múltiples direcciones *unicast*.

Aún más, a una sola interfaz se le pueden asignar direcciones múltiples. Cualquiera de las interfaces del nodo puede ser utilizada como un identificador del nodo. Todas las interfaces deben tener al menos una dirección *unicast* de *enlace local*.

Por la importancia que presentan los tipos de direcciones antes mencionados, a continuación se describen con detalle los mismos:

Direcciones Unicast.

Una dirección *unicast* es un identificador asignado a una sola interfaz. Así, un paquete enviado a una dirección *unicast*, solo será enviado a esa interfaz. Existen varias formas de asignación de direcciones *unicast*, algunas con estructuras más complejas que proporcionan asignación de direcciones jerárquicas.

Algunas direcciones *unicast* de propósito especial que han sido definidas hasta el momento de la escritura de este documento son las siguientes (en el apéndice A se describe de manera más detallada los tipos de direcciones):

- Dirección *loopback*. No debe ser asignada a una interfaz física, y todos los paquetes con esta dirección deben permanecer dentro del nodo creador del paquete. Aún más, los ruteadores no transmiten paquetes con direcciones *loopback*.
- Dirección no especificada. Esta dirección no puede ser asignada a una interfaz y no debe ser utilizada como dirección destino en los paquetes. Esta dirección indica la ausencia de una dirección IPv6, por ejemplo al inicializar un nodo en una red IPv6, se puede utilizar este tipo de direcciones como dirección fuente en los paquetes hasta que se reciba su dirección IPv6 del nodo mientras se realiza la *configuración automática*.
- Direcciones de host con *IPv6 con capacidad IPv4*.
 - Direcciones *IPv6 con compatibilidad IPv4*. Este tipo de direcciones tiene los 96 bytes de más alto orden con el valor “0” y los 32 bytes de más bajo orden de la dirección formados por la dirección IPv4 (A.B.C.D), y su formato es del tipo ::A.B.C.D. Este tipo de direcciones es utilizado como dirección IPv6 del nodo y la dirección incrustada IPv4 en los 32 bytes de más bajo orden es utilizada como dirección IPv4 del nodo.
 - Direcciones *IPv6 mapeadas a IPv4*. Este tipo de direcciones requiere que el host tenga habilitado Capa Dual IP (definido en la sección 1.7) o bien un ruteador tenga un traductor de encabezados.
- Direcciones de enlace local. Este tipo de direcciones pueden ser automáticamente configuradas en una interfaz utilizando el prefijo **FE80::/10**. También son

utilizadas en el proceso de *configuración automática* sin estados y en el protocolo de descubrimiento de vecinos MLD (*Multicast Listener Discovery*). Aún más, los ruteadores no transmiten paquetes con direcciones fuente o destino de enlace local a otros sitios.

- Direcciones de enlace de sitio. Utilizan el prefijo **FEC0::/10** y concatenan el identificador de subred con el identificador de interfaz. Éstas direcciones pueden ser utilizadas para ser asignadas a un sitio completo sin tener que utilizar una dirección de prefijo único global. Éstas direcciones pueden ser consideradas direcciones privadas debido a que pueden ser usadas para restringir la comunicación a un dominio limitado. Aún más, los ruteadores no transmiten paquetes con direcciones fuente o destino de este tipo fuera del sitio.
- Direcciones NSAP. Esta asignación de direcciones es utilizada para el protocolo NSAP (Network Service Access Point). Para realizar la traducción del DNS para el dominio *NSAP a nombre*, se utiliza el registro de recurso **RR**.
- Direcciones jerárquicas IPX. Esta asignación de direcciones es utilizada para el protocolo IPX (Internetwork Packet Exchange) de Novel.
- Direcciones *unicast* de alcance global. Es una dirección con el prefijo *unicast* de alcance global (**2000::/3**), la cual habilita el alcance estricto del prefijo de ruteo que limita el número de entradas en la tabla de ruteo global.

Los identificadores de interfaz en una dirección IPv6 son utilizados para identificar las interfaces en un enlace. Se requiere que sean únicos en un enlace. En algunos casos, un identificador de interfaz será el mismo que la dirección de la interfaz de la capa de enlace.

El uso del mismo identificador de interfaz en múltiples interfaces de un simple nodo, no afecta la unicidad del indentificador global de la interfaz.

Los detalles del indentificador de interfaz son definidos en la especificación apropiada “IPv6 sobre <el enlace>”, es decir, “IPv6 sobre Ethernet”, “IPv6 sobre FDDI”, etc. Es decir, un identificador de interfaz es utilizado para identificar interfaces en un enlace. El identificador de interfaz global *unicast* debe ser de 64 bytes de largo y construido en el formato modificado EUI-64.

Una dirección IPv6 con formato EUI-64 habilita el procesamiento IPv6 en una interfaz usando el identificador de interfaz EUI-64 en los 64 bytes de más bajo orden.

Direcciones Anycast.

Una dirección *anycast* es una dirección que es asignada a más de una interfaz, típicamente perteneciendo a distintos nodos, con la propiedad de que un paquete enviado a una dirección *anycast* es ruteado a la interfaz más cercana que tenga esa dirección, de acuerdo a la métrica de distancia del protocolo de ruteo. Las direcciones *anycast* tienen el formato que se muestra en la figura 1.3.

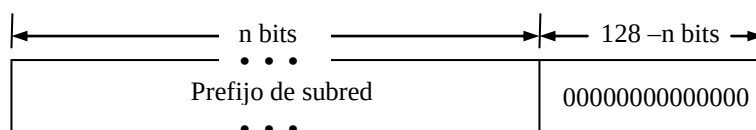


Figura 1.3. Formato de direcciones *anycast*.

Las direcciones *anycast* son localizadas en el espacio de direcciones *unicast*, utilizando cualquiera de los formatos de direcciones *unicast* definidos. Por tanto, una dirección *anycast* es indistinguible de una dirección *unicast*. Cuando una dirección *unicast* es asignada a más de una interfaz, entonces se convierte en una dirección *anycast*. Los nodos a los cuales la dirección es asignada deberán ser explícitamente configurados para saber que esta es una dirección *anycast*.

Algunos usos del esquema de direcciones *anycast* son los siguientes:

- Identificación de un conjunto de ruteadores pertenecientes a un proveedor de servicios de Internet.
- Identificación de un conjunto de ruteadores conectados a una subred particular.
- Identificación de un conjunto de ruteadores que proveen una entrada a un dominio particular de ruteo.

Algunas restricciones impuestas en el uso de direcciones *anycast*:

- Éstas no deben ser utilizadas como una dirección fuente para un paquete particular.
- Una dirección *anycast* solo debe ser asignada a un ruteador no a un host.
- Todos los ruteadores en una subred deben soportar las direcciones *anycast*.

Las direcciones *anycast* deberán ser utilizadas para aplicaciones donde un nodo necesita comunicación con un miembro de un grupo de ruteadores en una subred remota.

Direcciones *Multicast*.

Una dirección *multicast* identifica a un grupo de nodos y cada uno de estos nodos puede pertenecer a cualquier número de grupos *multicast*. El formato de una dirección *multicast* se presenta en la figura 1.4.

Los primeros 8 bits (todos uno's) de la dirección IPv6 identifican una dirección *multicast*. El campo “**flags**” es de cuatro bits, y los tres primeros bits de mas alto orden están reservados para uso futuro y deben ser inicializados a cero. El cuarto bit (el de menos orden) indica el tipo de dirección *multicast*, con T = 0 se trata de dirección permanente y con T = 1 es una dirección *multicast* temporal.

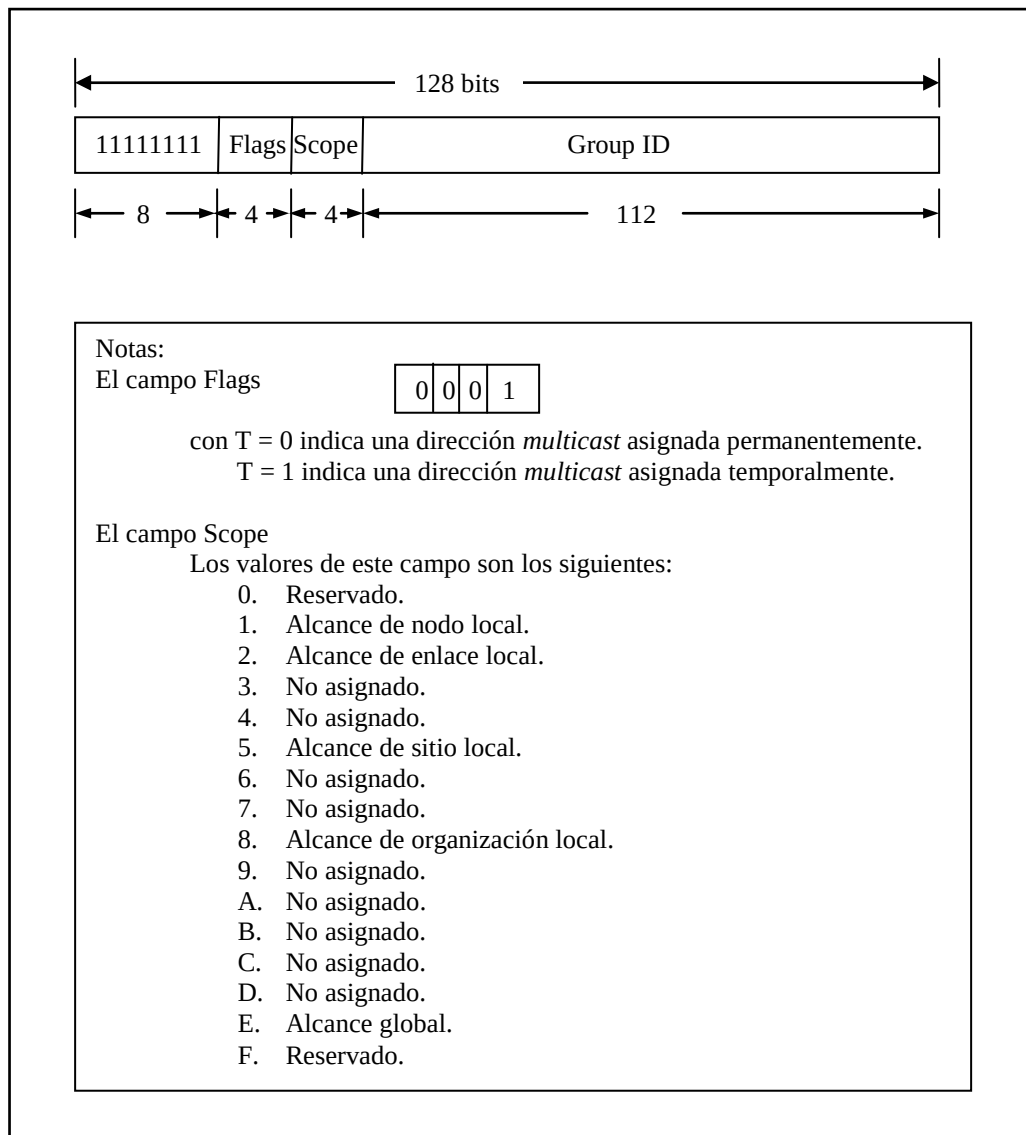


Figura 1.4. Formato de la dirección *multicast*.

El campo “*scope*” de cuatro bits es utilizado para limitar el alcance del grupo *multicast*.

El campo “*Group ID*” identifica el grupo *multicast*, ya sea permanente o temporal con el alcance dado en el campo “*scope*”.

Las direcciones *multicast* no deben ser utilizadas como una dirección fuente en el paquete IPv6 o aparecer en cualquier encabezado de ruteo. Existen varios tipos de direcciones *multicast* que están asignadas [HiR+98a].

1.3 Especificaciones de IPv6.

IPv6 fue diseñado para trabajar bien tanto en redes de alta tecnología como las redes *Gigabyt* y también es eficiente en redes *Ethernet*, *FDDI*, etc. Los cambios de IPv4 a IPv6 recaen principalmente en las siguientes categorías:

- Capacidades de direccionamiento expandidas. Aumento del tamaño de la dirección IP de 32 bits a 128 bits y simplicidad de auto configuración de la dirección.
- Formato de encabezado simplificado. Algunos campos del encabezado han sido eliminados, para hacerlos opcionales, con el fin de limitar el costo de ancho de banda y reducir el costo de procesamiento de manejo de paquetes en algunos casos particulares.
- Soporte mejorado para extensiones y opciones. Gran flexibilidad para introducir nuevas opciones en el futuro.
- Capacidad de etiquetamiento de flujo. Permite etiquetar paquetes que pertenecen a un flujo de tráfico particular, para el cual la fuente solicita manejo especial, tal como *servicio en tiempo real* o *QoS no default*.
- Capacidades de autenticación y privacidad. Extensiones para soportar autenticación, integridad de datos y confidencialidad son especificadas en IPv6.

El formato del encabezado de un paquete IPv6 ha sido simplificado de su contraparte IPv4. La longitud del encabezado IPv6 se ha incrementado a 40 bytes (de los cuales cuando menos 20 bytes son de IPv4) y contiene dos direcciones (fuente y destino) de 16 bytes cada una precedidas por 8 bytes de información de control, como se muestra en la figura 1.2, comparado con el encabezado IPv4 que tiene dos direcciones de 4 bytes precedidas por 12 bytes de información y posiblemente seguidos por otros bytes de opción de datos.

La reducción de la información de control y la eliminación de opciones en el encabezado para muchos paquetes es con la intención de optimizar el tiempo de procesamiento por paquete en un ruteador. Los campos poco utilizados en IPv4 se han eliminado y convertido en extensiones de encabezado opcionales cuando sean requeridos.

Este encabezado básico será en muchos casos el único encabezado necesario para enviar el paquete, y cuando sea necesario llevar información adicional junto con el paquete al destino o a un nodo intermedio en la ruta se utilizarán las extensiones de encabezado adecuadas. Éstas son colocadas inmediatamente después del encabezado básico y son contadas como parte de la carga del paquete.

Esta estructura permite a IPv6 encadenar múltiples extensiones de encabezado como sea necesario (en el apéndice A se presenta información más detallada sobre el tema).

1.4 Tamaño del paquete.

El diseño de IPv6 define restricciones sobre el tamaño mínimo y máximo de un paquete [KeS+98a]. Para cada enlace al cual el nodo esta directamente conectado, el nodo deberá ser capaz de aceptar paquetes tan grandes como su enlace MTU.

El tamaño mínimo de un paquete en un enlace IPv6 en Internet es de 1,280 bytes, de manera similar, para enlaces con una MTU configurable, tales como enlaces PPP (Point to Point Protocol) el mínimo MTU configurable deberá ser 1,280 bytes, 1,500 bytes es recomendable.

El tamaño máximo de un paquete con el encabezado IPv6 incluida es de 65,575 bytes, por lo tanto, el tamaño máximo del paquete (datos mas extensiones de encabezado) debe ser de 65,535 bytes. Para una longitud de carga mayor a 65,535 bytes, la opción “**Jumbo Payload**” junto con la opción “**Hop-by-Hop**” deberá ser habilitada. Así es posible enviar una carga de datos entre 65,535 bytes y 4,294’967,295 bytes. Esto solo es posible realizarlo en redes “**Gigabyte**”, “**ATM Gigabyte**” es una de ellas.

IPv6 fue diseñado para que todos los nodos dinámicamente determinen la máxima MTU soportada por todos los enlaces a lo largo de la trayectoria [MoJ+90]. Así, los nodos fuente solo enviarán paquetes que no excedan la MTU de la trayectoria. Esta es la razón de que los ruteadores IPv6 no tengan que fragmentar paquetes a la mitad de la trayectoria, para así conseguir un más eficiente uso de las trayectorias que atraviesen diversos medios de transmisión. Además, el protocolo IPv6 requiere que cada enlace soporte un MTU de 1280 bytes o mayor.

1.5 Requerimientos de direcciones para los nodos.

Los requerimientos mínimos que un host debe satisfacer son:

1. Un host debe reconocer las siguientes direcciones así como la identificación de si mismo:
 - Su dirección de enlace local para cada interfaz.
 - La dirección *unicast* asignada.
 - La dirección *loopback*.
 - Las direcciones *multicast* de todos los nodos.
 - Las direcciones *multicast* a solicitud del nodo para cada una de sus direcciones *unicast* y *anycast*.
 - Dirección *multicast* de todos los otros grupos a los cuales el host pertenece.

2. Un ruteador debe reconocer las siguientes direcciones así como la identificación de si mismo:
 - Las direcciones *anycast* de la subred del ruteador para que las interfaces que estén configuradas actúen como ruteadores.
 - Todas las demás direcciones *anycast* con las cuales el ruteador ha sido configurado.
 - Las direcciones *multicast* de todos los ruteadores.
 - Las direcciones *multicast* a solicitud del nodo para cada una de sus direcciones *unicast* y *anycast*.
 - Las direcciones *multicast* de todos los otros grupos a los cuales el ruteador pertenece.

1.6 Compatibilidad de direcciones.

Existen dos tipos de direcciones las cuales han sido definidas para la transición de redes IPv4 a IPv6.

1. *La dirección IPv6 con compatibilidad IPv4*, la cual es utilizada cuando dos dispositivos IPv6 tales como host o ruteadores necesitan comunicarse vía una infraestructura de ruteo IPv4. estos dispositivos interconectados a la red IPv4 deberán utilizar una dirección *unicast* especial, la cual lleva la dirección IPv4 en los últimos 32 bits de más bajo orden. Este proceso es llamado **túnel automático** y tiene el formato ilustrado en la figura 1.5.
2. *La dirección IPv6 mapeada a IPv4*, esta dirección es utilizada por nodos que soportan IPv4 solamente y no soportan IPv6. Esto es, cuando un host IPv6 necesita comunicarse con un host que solo soporta IPv4, utiliza este tipo de direcciones. En la figura 1.6 se ilustra este formato de direcciones.

Los nodos IPv6/IPv4 pueden utilizar mecanismos de configuración de dirección IPv6 sin estados o DHCP para adquirir su dirección.

Para IPv6, la jerarquía de las direcciones *incorporables* (*Aggregatable Address*), es organizada en tres niveles [HiR+98b]:

- Topología pública. Es la colección de proveedores de servicios de Internet (ISP Internet Service Providers) que proporcionan un servicio de tránsito de Internet público.
- Topología de sitio. Es local a un sitio específico u organización, pero no proporciona servicio de tránsito de Internet público a nodos fuera del sitio.
- Identificador de interfaz. Proporciona una identificación única a las interfaces en un enlace específico.

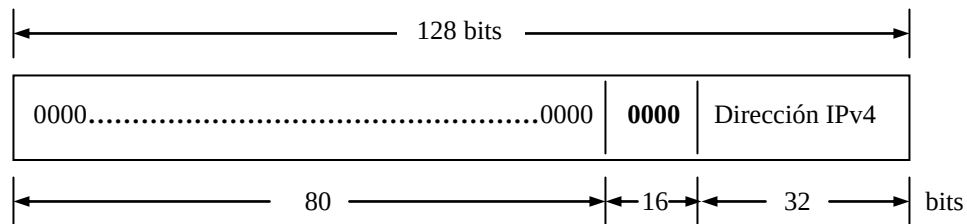


Figura 1.5. Formato de direcciones *IPv6 con compatibilidad IPv4*.

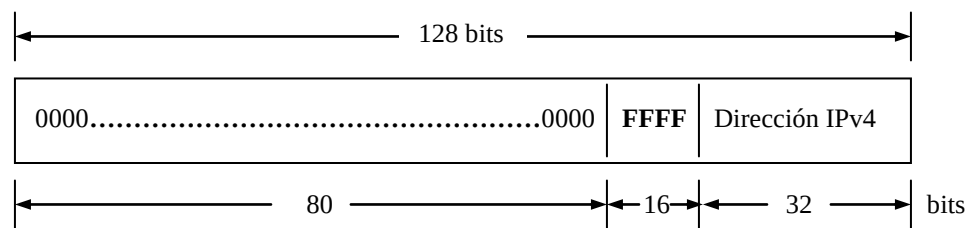


Figura 1.6. Formato de direcciones *IPv6 mapeadas a IPv4*.

1.7 El futuro de IPv6.

Con un esquema de direccionamiento tan grande, es posible utilizar algunas de las direcciones para hacer la administración de la red y el ruteo más fácil. Este nuevo esquema de direccionamiento, permite configuración de direcciones automática y reconfiguración, es decir, los servidores pueden ser reenumerados con direcciones de red (cambiados de IP) sin que todos los clientes que accedan y clientes móviles tengan que moverse sobre la red.

Las especificaciones de IPv6 tienen muchas API's (Application Program Interfaces) para habilitar la comunicación IPv6, y muchas son independientes de la versión IP utilizada y el más común API es el **socket básico API**. De esta forma, los desarrolladores pueden escribir segmentos de código que soporten ambos protocolos de comunicación IPv4 e IPv6.

Basados en el nombre del sistema que es el emisor de la comunicación, y la configuración del nodo actual, la API determina si la dirección IP está utilizando el protocolo IPv4 ó IPv6. Así, utilizando una versión IP independiente del API, los desarrolladores pueden habilitar la comunicación transparentemente. Una vez que la

aplicación tiene un puerto hacia la interfaz, el código del socket determina, basado en la configuración actual, si la comunicación utiliza el protocolo IPv4 ó IPv6 y por tanto, la aplicación decide automáticamente cual versión del protocolo seleccionar. Esto habilita el uso de IPv4 por ahora y requiere que una nueva codificación utilice IPv6 más adelante.

Los servidores de nombres deberán administrar ambientes IPv4 e IPv6. Una vez que un servidor de nombres ha sido actualizado para soportar IPv6, este regresará la dirección apropiada (IPv4 ó IPv6) para el sistema remoto basado en las capacidades y configuración del ambiente. Utilizando el servidor de nombres para administrar la información sobre IPv4 e IPv6 hace el desarrollo del código y la administración mucho más fácil para los desarrolladores.

IPv6 es el futuro de Internet, y será necesario esperar algún tiempo para ver si IPv6 es el protocolo estándar del futuro. Además, aunque los desarrolladores accedan a nuevas capacidades a través de nuevas interfaces y aunque algunos parámetros tales como la dirección del nodo tengan que cambiar entre las dos versiones, la API misma deberá cambiar. Esta es la tarea de los desarrolladores para hacer esta transición lo más simple posible.

Hasta el momento, existe una cierta duda, sobre el tiempo que será necesario esperar para que Internet se convierta a IPv6 solamente. Por esta razón, el socket básico API soporta IPv4 e IPv6. Esta aproximación es llamada **Dual Stack Interface**, como se ilustra en la figura 1.7. Una vez que la aplicación ha sido actualizada a la interfaz de socket IPv6, ya no habrá necesidad de utilizar el código requerido para habilitar la comunicación con ambos protocolos IPv4 e IPv6.

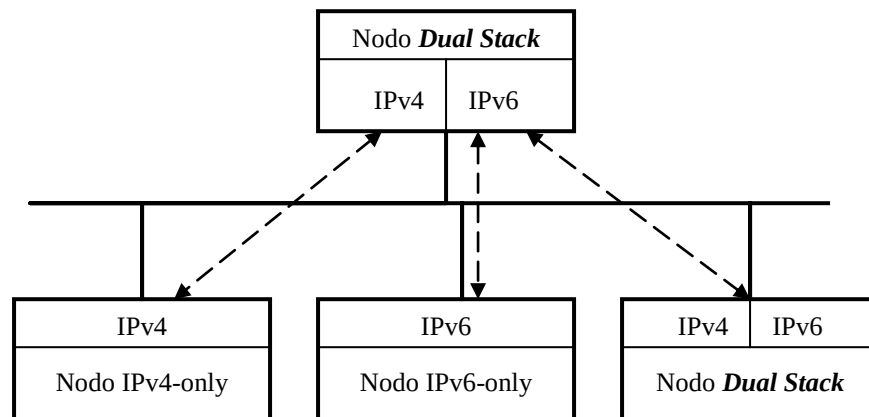


Figura 1.7. Posibilidades de enlace de la interfaz de *Capa Dual IP*.

En la tabla 1.3, se ilustra la forma en que interactúan ambos protocolos [SUN99]. En esta tabla “X” denota que la comunicación entre el servidor respectivo y el cliente no es posible en la capa IP. Otras herramientas como *proxies*, pueden ser utilizados para este

tipo de comunicación. La tabla 1.3, supone que la interfaz de *Capa Dual IP* tiene ambas direcciones IPv4 e IPv6 en el correspondiente servidor de nombres. Un nodo *IPv6-unaware* tiene solo una *Capa Dual IPv4* como medio de comunicación. Un nodo *IPv6-enabled* tiene una *Capa Dual IP* y a lo menos una interfaz IPv6 configurada. En un nodo de tipo *IPv6-only* el sistema implementa solamente el stack IPv6 (capa IP) y solo tiene una dirección en la base de datos del servidor de nombres.

Type of Application. (Type of Node)	IPv6-unaware Server (IPv4-only Node)	IPv6-unaware Server (IPv6-enabled Node)	IPv6-aware Server (IPv6-only Node)	IPv6-aware Server (IPv6-enabled Node)
IPv6-unaware Client (IPv4-only Node)	IPv4	IPv4	X	IPv4
IPv6-unaware Client (IPv6-enabled Node)	IPv4	IPv4	X	IPv4
IPv6-aware Client (IPv6-only Node)	X	X	IPv6	IPv6
IPv6-aware Client (IPv6-enabled Node)	IPv4	(IPv4)	IPv6	IPv6

Tabla 1.3. Interoperabilidad IPv4 e IPv6.

Existen otros métodos para que las aplicaciones se comuniquen, como lo son las RPC's (Remote Procedure Calls). Esta facilidad es construida en una base independiente del protocolo. Utilizando esta interfaz, los desarrolladores de aplicación pueden utilizar éstas llamadas para comunicarse transparentemente con otros clientes IPv6.

CAPÍTULO 2 Mecanismos de migración.

2.1 Perspectiva de transición.

La clave para el éxito de la migración hacia IPv6, es la compatibilidad con host y ruteadores que actualmente manejan IPv4. Así, manteniendo la compatibilidad con IPv4 mientras se desarrolla IPv6, hará la transición más fácil y menos traumática.

Una de las características de IPv6 es el soporte para el encapsulamiento de si mismo y de otros protocolos, además, los métodos de transición existentes para la migración de IPv4 hacia IPv6 son compatibles con los métodos existentes en IPv4.

La transición de IPv4 a IPv6 tendrá que satisfacer algunos requisitos: Un requisito a corto plazo tendrá que definir las tecnologías específicas y métodos de transición en redes IPv4 y el Internet de hoy hacia redes IPv6 y el Internet del futuro. Un requisito a largo plazo será lograr una planeación y estrategias operacionales en redes basadas en banda ancha (principalmente), desarrollando métodos que permitan guiar las estrategias de migración, entendiendo que tiene que existir una enorme interacción por un largo periodo de coexistencia ya que ambos protocolos son parte de la infraestructura básica. Otra característica fundamental de estas estrategias de migración, tendrá que lograr que el Internet del futuro, es decir IPv6 sea seguro, confiable y fácil de administrar.

La forma más directa de que un nodo IPv6 pueda permanecer compatible con IPv4 es proporcionar una implementación completa IPv4.

Existen algunas estrategias de migración diseñadas para propósitos específicos. En este documento, se describirán con detalle algunos mecanismos comunes que sirvieron de base para desarrollar la estrategia de *tunneling* que aquí se presenta. Otros mecanismos existentes están fuera del alcance de este documento.

Existen cuatro criterios básicos de transición [BrS+95], los cuales son:

- Actualización incremental. Permitir la existencia de host IPv4 para que sean actualizados en cualquier momento, sin depender de que otros host o ruteadores sean actualizados.
- Desarrollo incremental. Los nuevos host y ruteadores IPv6 deben ser instalados en cualquier momento sin requisitos previos.
- Fácil direccionamiento. Cuando existan host y ruteadores IPv4 instalados y que sean actualizados a IPv6, estos podrán mantener sus direcciones IPv4 existentes sin necesidad de asignarles nuevas direcciones IPv4.
- Bajo costo de cambio. Poco o nulo trabajo deberá realizarse con objeto de actualizar los sistemas IPv4 existentes a IPv6, o bien desarrollar nuevos sistemas IPv6.

Dependiendo en que fase de la transición se encuentre, serán los pasos a realizar, tipo de configuraciones, actualización de sistemas, etc.

En la siguiente sección se definen algunos de los mecanismos existentes de migración, describiendo sus características sobresalientes y en la sección 2.3 se enfoca la atención en los tipos de túneles, realizando una comparación entre los mismos.

2.2 Tunneling.

El *tunneling* es un proceso donde la información de un protocolo es encapsulada dentro de la trama o paquete de otra arquitectura, habilitando así que los datos originales sean llevados sobre otra arquitectura. Los escenarios de *tunneling* para IPv4 hacia IPv6 son diseñados para habilitar a una infraestructura IPv4 existente a llevar paquetes IPv6 mediante el encapsulamiento de la información dentro de datagramas IPv4 [MiM99].

También se puede describir al *tunneling* IPv6 como una técnica para establecer un *enlace virtual* entre dos nodos IPv6 para transmitir paquetes de datos como carga de los paquetes IPv6. Desde el punto de vista de los dos nodos, este *enlace virtual* llamado *túnel IPv6* aparece como un enlace punto a punto en el cual IPv6 actúa como un protocolo de la capa de enlace del modelo referencia OSI (Open Systems Interconnection).

Antes de entrar en materia sobre el tema, se definen algunos términos que relacionan a nodos y direcciones para uso en los mecanismos de transición y que serán de utilidad en el resto del documento [GiR+96]. Uno o más de estos mecanismos deberán ser implementados por los host y ruteadores IPv6 con objeto de mantener compatibilidad con los host y ruteadores que actualmente manejan IPv4, considerando que la instalación inicial de host y ruteadores IPv4 existente antes de la transición son nodos que solamente manejan IPv4.

- Tipos de nodos.
 - Nodo *IPv4-only*. Un host o un ruteador que implementa solo IPv4. Un nodo *IPv4-only* no entiende IPv6.
 - Nodo *IPv6-only*. Un host o un ruteador que implementa IPv6 y no implementa IPv4.
 - Nodo IPv6/IPv4. un host o un ruteador que implementa ambos IPv4 e IPv6.
 - Nodo IPv6. Un host o ruteador que implementa IPv6. Los nodos IPv6/IPv4 e *IPv6-only* son ambos nodos IPv6.
 - Nodo IPv4. Un host o ruteador que implementa IPv4. Los nodos IPv6/IPv4 e *IPv4-only* son ambos nodos IPv4.
- Tipos de direcciones.
 - Dirección IPv6 con compatibilidad IPv4. Una dirección IPv6 asignada a un nodo IPv6/IPv4, el cual asigna el prefijo de los 96 bits de más alto orden, es

decir, 0 : 0 : 0 : 0 : 0 : 0, y una dirección IPv4 en los 32 bits de mas bajo orden. Las direcciones IPv6 con compatibilidad IPv4 son utilizadas por el mecanismo de tunneling automático.

- Dirección *IPv6-only*. Son el resto del espacio de direcciones IPv6. Es decir, las direcciones que se les asigna otro prefijo distinto a 0:0:0:0:0:0.
- Existen algunas técnicas usadas en la transición, entre las cuales se encuentran las siguientes:
 - Capa dual IP. Deberá proveer soporte completo para ambos protocolos (IPv4 e IPv6), en host y ruteadores (definido en la sección 1.7).
 - Tunneling *IPv6-over-IPv4*. Es la técnica de encapsulamiento de paquetes IPv6 dentro de paquetes IPv4, tales que estos puedan ser llevados sobre una infraestructura de ruteo IPv4. En esta técnica se describen dos alternativas de tunneling, configurado y automático.
 - Tunneling configurado. Es la técnica de tunneling *IPv6-over-IPv4* donde la dirección de destino del túnel IPv4 es determinada por la información de configuración del nodo encapsulante. El túnel puede ser unidireccional o bidireccional (enlace punto a punto).
 - Tunneling automático. Es la técnica de tunneling *IPv6-over-IPv4* donde la dirección de destino del túnel IPv4 es determinada de la dirección IPv4 insertada en la dirección de destino IPv6 con compatibilidad IPv4 del paquete IPv6.
 - Tunneling 6to4. Es una conexión de islas IPv6 sin un túnel específico. Esta técnica provee una solución al problema complejo de utilizar túneles manualmente configurados, especificando un único prefijo de ruteo para cada usuario final que lleve una dirección destino IPv4 en el túnel.

Estas definiciones son consideradas esenciales, y se ampliarán un poco para comprender adecuadamente el concepto.

Capa Dual IP

La técnica de *Capa Dual IP* mencionada anteriormente, puede o no ser utilizada en conjunción con las técnicas de encapsulamiento *IPv6-over-IPv4*. Un nodo *IPv4 to IPv6* que soporta *tunneling* puede soportar solamente *tunneling* configurado o ambos, *tunneling* configurado y *tunneling* automático. Así se pueden realizar tres configuraciones posibles:

- Un nodo IPv6/IPv4 que no realice *tunneling*.
- Un nodo IPv6/IPv4 que realice *tunneling* configurado solamente.
- Un nodo IPv6/IPv4 que realice *tunneling* configurado y *tunneling* automático.

Los nodos IPv6/IPv4 pueden ser configurados con ambas direcciones IPv6 e IPv4, aunque las dos direcciones pueden estar relacionadas una con otra, esto no es necesario.

Los nodos *IPv4 to IPv6* pueden estar configurados con direcciones IPv4 e IPv6 que no estén relacionadas una con la otra.

Tunneling IPv6-over-IPv4

Los nodos que realizan *tunneling* automático son configurados con direcciones *IPv6 con compatibilidad IPv4*. Estos pueden ser vistos como una simple dirección que puede servir a ambos como dirección IPv4 y dirección IPv6. Así, los 128 bits de la dirección *IPv6 con compatibilidad IPv4* son utilizados como la dirección IPv6 del nodo mientras que la dirección IPv4 insertada en los 32 bits de más bajo orden sirve como la dirección IPv4 del nodo.

En el caso de la dirección *loopback*, los paquetes no deberán salir del nodo. La implementación IPv6/IPv4 trata a esta dirección *IPv6 con compatibilidad IPv4 ::127.0.0.1* como una dirección *loopback* IPv6.

Aún teniendo la compatibilidad de direcciones, el soporte actual para el almacenamiento de direcciones de Internet en el DNS (*Domain Name System*) no puede ser fácilmente extendido a las direcciones IPv6 ya que muchas de las aplicaciones suponen que la consulta de la dirección regresa una dirección IPv4 de 32 bits.

El soporte para el almacenamiento de direcciones IPv6 define las siguientes extensiones:

- Un nuevo tipo de registro para convertir un nombre de dominio en una dirección IPv6 [ThS+95].
- Un nuevo dominio para soportar “lookup” basado en direcciones.
- Existen consultas que realizan procesamiento adicional para localizar una dirección IPv4, éstas son redefinidas para realizar procesamiento adicional en ambas direcciones IPv4 e IPv6.

Estos cambios son diseñados para ser compatibles con el software existente. El soporte existente para direcciones IPv4 es mantenido.

Este nuevo tipo de registro es definido para almacenar una dirección IPv6 de un host. Un host que tiene más de una dirección IPv6 debe tener más de uno de tales registros. Este tipo de registro del recurso “AAAA” es un nuevo registro específico para la clase de Internet que almacena una sola dirección IPv6. La dirección IPv6 de 128 bits es codificada en la porción de datos de un registro de recurso en el orden de bytes de red.

Cuando una dirección *IPv6 con compatibilidad IPv4* es asignada a un host IPv6/IPv4 que soporta un túnel automático, ambos registros “A” (de IPv4) y “AAAA” (de IPv6) son listados en el DNS. El registro “AAAA” mantiene la dirección completa *IPv6 con compatibilidad IPv4* (128 bits), mientras que el registro “A” mantiene los 32 bits de orden más bajo de la dirección.

Cada dominio puede tener un conjunto asociado de registros de recurso, además el procedimiento de resolución realmente recibe registros de recurso del servidor DNS.

Una característica importante de estos túneles, es que estos son modelados como un *simple salto*. Esto es, el límite de saltos en IPv6 es decrementado por uno cuando un paquete atraviesa el túnel. El modelo *simple salto* es oculto a el usuario y aún más no es detectado por herramientas tales como *traceroute*, además este modelo sirve para ocultar la presencia del túnel.

Este modelo debe ser implementado por los nodos de entrada y salida del túnel (nodo encapsulador y desencapsulador) para procesar el campo *limit hop* de IPv6, y si transmiten el paquete decrementan por uno el valor de este campo. El nodo fuente y el nodo destino no decrementan el campo *limit hop*.

Tunneling 6to4

El *tunneling 6to4* es una técnica donde el punto final del túnel es determinado por la dirección IPv4 única globalmente empujada en la dirección IPv6. Una dirección IPv4 es una combinación de un prefijo único de ruteo **2002::/16** y una dirección IPv4 (de 32 bytes) única globalmente. Las direcciones *IPv6 con compatibilidad IPv4* tienen un formato diferente de las direcciones *6to4*, y las direcciones *IPv6 con compatibilidad IPv4* no son utilizadas en el *tunneling 6to4*.

Los túneles 6to4 son configurados en los ruteadores de frontera o entre los ruteadores y un host. Aún más, es necesario configurar un ruteador de frontera utilizando *Capa Dual IP* que sea el punto final del túnel *6to4*.

Ahora que se tiene el concepto del tipo de nodos, direcciones y túneles, se describirá el papel que juegan los dos nodos IPv6 en el proceso de *tunneling*.

- El nodo encapsulador agrega un encabezado al paquete original recibido de otros nodos o de si mismo.
- El nodo desencapsulador quita ese encabezado del paquete recibido y transmite el paquete original hacia su destino, posiblemente el mismo.
- El nodo encapsulador es llamado punto de entrada del nodo túnel y esta es la dirección fuente del encabezado agregado al paquete encapsulado.
- El nodo desencapsulador es llamado punto de salida del nodo túnel y esa es la dirección destino del encabezado agregado al paquete encapsulado.

El proceso de *tunneling* se refiere al *tunneling* entre dos nodos identificados por una dirección *unicast*. En las figuras 2.1 y 2.2 se ilustra este proceso que involucra los siguientes pasos:

1. Encapsulamiento. En el nodo (punto de entrada del túnel) el encabezado IPv4 es creado y el paquete encapsulado es transmitido. Este nodo también puede mantener la información de la configuración con respecto a los túneles que son establecidos, tales como el tamaño de la MTU que es soportada en el túnel.
2. Desencapsulamiento. En el nodo (punto de salida del túnel) el encabezado IPv4 es quitado y el paquete IPv6 es procesado.
3. Administración del túnel. Es la configuración del túnel.

Los host y ruteadores pueden encapsular datagramas IPv6 sobre regiones de la topología de ruteo IPv4, encapsulando estos sobre paquetes IPv4.

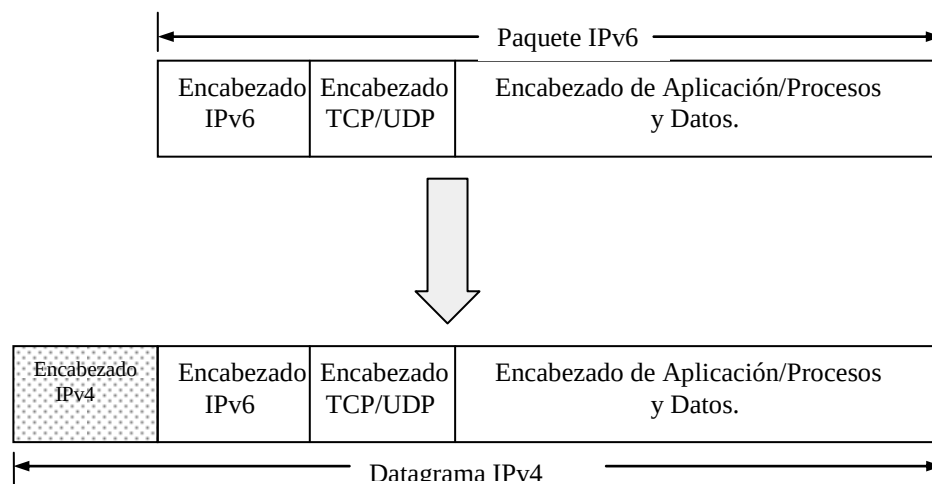


Figura 2.1. Proceso de encapsulamiento.

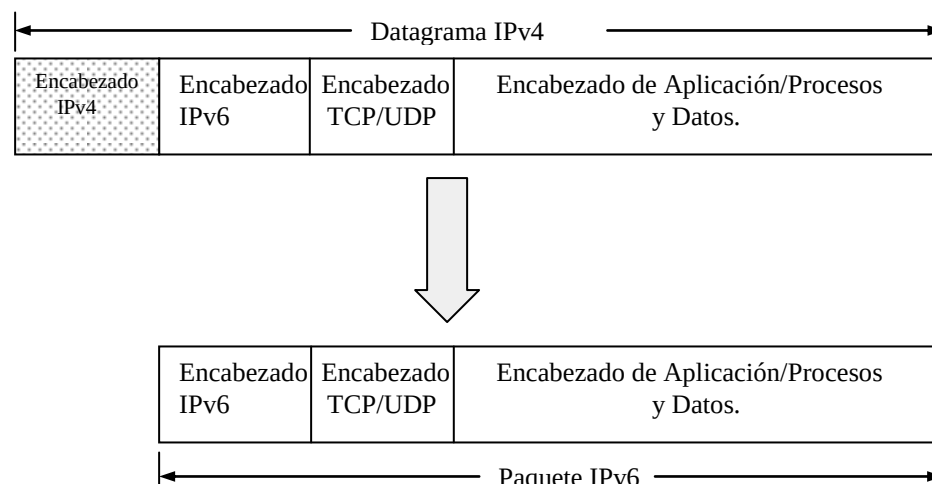


Figura 2.2. Proceso de Desencapsulamiento.

Hasta aquí, se han descrito de manera general las dos técnicas utilizadas en la transición, *tunneling IPv6-over-IPv4* y *tunneling 6to4*. Ahora nos falta describir con

detalle los dos tipos de túneles existentes en la técnica de *tunneling IPv6-over-IPv4*, los cuales son *tunneling automático* y *tunneling configurado*, esto se presenta en la siguiente sección.

2.3 Tipos de túneles.

Se definen cuatro posibles configuraciones de túneles [GiR+96] que podrán ser establecidas entre ruteadores y host y se ilustran en la figura 3. En el apéndice B se da una descripción más detallada al respecto. Estas configuraciones son:

- *Ruteador a Ruteador*. Los ruteadores IPv6/IPv4 que están separados por una infraestructura IPv4, pueden encapsular paquetes IPv6 entre ellos mismos, en este caso el túnel abarca un segmento de la trayectoria *punto a punto* que el paquete IPv6 toma. Esta categoría recae en el tipo de túneles automático.
- *Host a Ruteador*. Un host IPv6/IPv4 puede encapsular paquetes IPv6 a un ruteador intermedio IPv6/IPv4 que es accesible vía una infraestructura de ruteo IPv4. este tipo de túnel abarca el primer segmento de la trayectoria *punto a punto* del paquete. Esta categoría recae en el tipo de túneles automático.
- *Host a Host*. Los host IPv6/IPv4 que están interconectados por una infraestructura IPv4 pueden encapsular paquetes IPv6 entre ellos mismos. En este caso el túnel abarca la trayectoria *punto a punto* completamente que el paquete toma. Esta categoría recae en el tipo de túneles configurado.
- *Ruteador a Host*. Los ruteadores IPv6/IPv4 pueden encapsular paquetes IPv6 a su destino final, el host IPv6/IPv4. este túnel solo abarca el último segmento de la trayectoria *punto a punto*. Esta categoría recae en el tipo de túneles configurado.

Estas cuatro configuraciones posibles de encapsulamiento, se dividen en dos tipos de túneles, los cuales se describen a continuación.

Túnel configurado.

En la figura 2.3a muestra la configuración del túnel *ruteador a ruteador*. La dirección destino del túnel IPv4 es determinada por la información de configuración del túnel del nodo encapsulante.

Este tipo de túnel puede ser unidireccional o bidireccional. Nótese que la dirección de destino del túnel es diferente de la dirección de destino final. La dirección final de destino deberá ser la del host no la del ruteador.

Se dice que es túnel configurado debido a que la dirección del punto de salida del túnel y la dirección destino del host difieren, el nodo que realiza el encapsulamiento determina el destino del túnel (en este caso es el ruteador) de alguna información de la configuración del mismo.

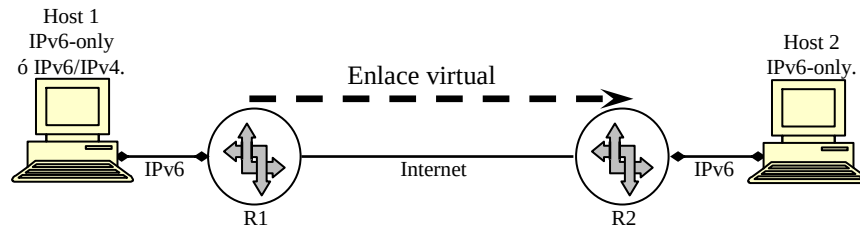


Figura 2.3a. Túnel configurado de router a router.

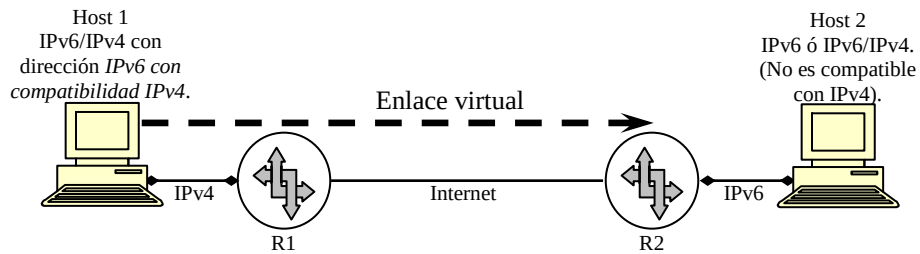


Figura 2.3b. Túnel configurado de host a router.

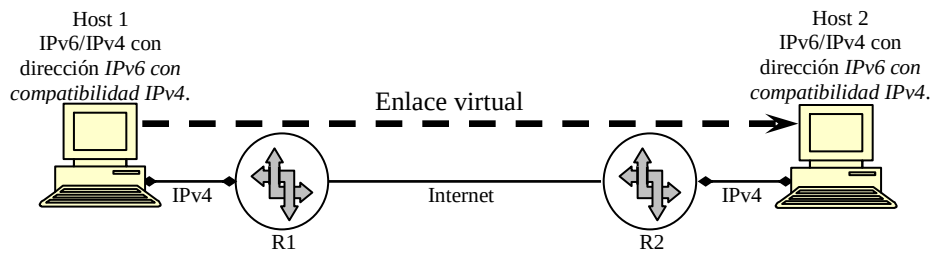


Figura 2.3c. Túnel automático de host a host.

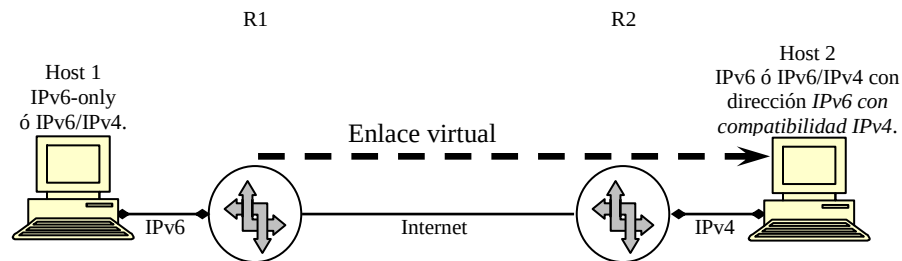


Figura 2.3d. Túnel automático de router a host.

Figura 2.3. Configuraciones de encapsulamiento existentes.

Otra forma de interconexión se muestra en la figura 2.3b, donde el host de origen es configurado como punto de entrada del túnel. En este caso el host fuente debe ser un host dual (soportando ambos protocolos) y el túnel debe ser configurado entre el host 1 y el ruteador 2. El encapsulamiento ocurre en el host 1 (fuente) y el desencapsulamiento en el ruteador 2 que es el punto de salida del túnel. El host 2 no tiene una dirección *IPv6 con compatibilidad IPv4*.

Túnel automático.

Este tipo de túnel se define como un encapsulamiento *IPv6-over-IPv4* donde la dirección destino del túnel IPv4 es determinada utilizando la dirección IPv4 agregando esta última en los 32 bits de más bajo orden de la dirección destino *IPv6 con compatibilidad IPv4* del paquete que esta siendo encapsulado.

De los escenarios presentados en esta sección, el *host a host* y *Ruteador a host*, ambos terminan en un *host*, además, otro detalle importante es que la dirección del punto de salida del túnel y la dirección del destino del paquete IPv6, identifican el mismo dispositivo (el host). Aún más, si las direcciones IPv4 e IPv6 pueden ser correlacionadas, entonces el proceso de *tunneling* es simplificado.

Esta correlación considerablemente simplificada proporciona facilidad para asignar las direcciones *IPv6 con compatibilidad IPv4*. En la figura 2.3c se ilustra la estructura de las direcciones *IPv6 con compatibilidad IPv4* y en la cual, la dirección IPv4 ocupa los 32 bits de más bajo orden, y el resto es llenado de ceros. Ya que la dirección destino puede ser fácilmente derivada de la dirección *IPv6 con compatibilidad IPv4*, este tipo de túnel es llamado túnel automático.

En esta misma figura (2.3c) se muestra la manera en que dos host IPv6 se comunican sobre una infraestructura de ruteo IPv4 usando el túnel automático. El host fuente genera y encapsula el paquete, y las direcciones asignadas son la de el mismo y la del destinatario (host 1 y host 2), estas direcciones son fácilmente derivadas de las direcciones conocidas *IPv6 con compatibilidad IPv4* eliminando los ceros de la izquierda (96 bits). Este proceso de derivación de direcciones automáticamente da el punto de entrada del túnel y el punto de salida del túnel.

De manera similar en la figura 2.3d se ilustra un ruteador dual, el cual recibe los paquetes destinados para un host con dirección *IPv6 con compatibilidad IPv4* y automáticamente los encapsula al destino final (host 2).

El encapsulamiento automático *host a host* permite a IPv6 desarrollarse en los host sin tener que cambiar la infraestructura de ruteo.

Las técnicas de encapsulamiento “configurado y automático” difieren principalmente en el método que utilizan para determinar la dirección de salida del túnel. Muchos de los mecanismos intrínsecos son los mismos:

- El nodo de entrada del túnel (nodo encapsulante) añade un encabezado IPv4 y transmite el paquete encapsulado.
- El nodo de salida del túnel (nodo desencapsulante) recibe el paquete encapsulado, quita el encabezado IPv4, actualiza el encabezado IPv6 y procesa el paquete IPv6 recibido.
- El nodo encapsulante puede necesitar mantener información breve para cada túnel registrando parámetros como la MTU del túnel con objeto de procesar los paquetes IPv6 y transmitirlos en el túnel.

Existen otros tipos de túneles de tal forma que la dirección de destino de salida del túnel es encontrada por otros medios, algunos de estos se describen a continuación.

Túnel *multicast*.

Multicast Tunneling. Es definido como un túnel IPv6 sobre IPv4 donde la dirección de salida del túnel IPv4 es determinada utilizando el protocolo de descubrimiento de vecinos MLD.

A diferencia del túnel configurado, este no requiere cualquier configuración de la dirección y a diferencia del túnel automático, este no requiere el uso de direcciones *IPv6 con compatibilidad IPv4*, lo cual requiere que la infraestructura para IPv4 soporte *IPv4 multicast*.

Túnel combinado.

Es una combinación de algunas de las técnicas de encapsulamiento definidas anteriormente.

Túnel genérico.

Son mecanismos que permiten llevar carga útil de una variedad de diferentes protocolos. Ejemplos: AppleTalk, IPX, CLNP, etc.

Los escenarios de transición ilustran los escenarios de ruteo para transmitir paquetes entre dos regiones.

2.3.1 Mecanismos comunes de los túneles.

Además de agregar el encabezado IPv4, el nodo encapsulante deberá realizar otras acciones:

- Determinar cuando fragmentar y cuando reportar un error ICMP de paquete demasiado grande al nodo fuente.
- Como tratar los errores ICMP IPv4 de los ruteadores junto con la trayectoria recorrida en el túnel hacia la fuente y los errores ICMP IPv6.
- Configurar adecuadamente las direcciones de entrada y salida del túnel.

Cuando se desencapsula un paquete IPv6/IPv4, el encabezado IPv6 no es modificado. Si el paquete es subsecuentemente transmitido, el campo *hop limit* es decrementado por uno.

Después de que el paquete IPv6 es desencapsulado, este es procesado como cualquier otro paquete IPv6 recibido.

Un túnel IPv6 es un mecanismo unidireccional. Esto es, un paquete fluye en una dirección de un punto de origen a un punto de destino al ser encapsulado.

El túnel bidireccional es logrado construyendo dos mecanismos de túnel unidireccionales, esto es, configurando dos túneles cada uno opuesto en dirección al otro. Así el punto de entrada del túnel en un nodo es el punto de salida del túnel en el otro nodo.

En la figura 2.4 se ilustra un mecanismo de túnel bidireccional. En un túnel bidireccional, se pueden tener túneles unidireccionales configurados independientemente, esto es, se pueden tener dos mecanismos de túnel utilizando técnicas diferentes de encapsulamiento.

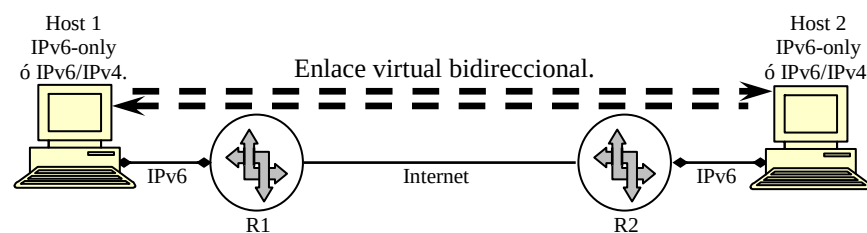


Figura 2.4. Mecanismo de Túnel bidireccional.

2.4 Estrategias de diseño.

Al haber analizado brevemente las técnicas de *tunneling* y los tipos de túneles, se puede deducir cuales son las características que nos interesan de cada una de estas

técnicas para así iniciar el diseño de la técnica de *tunneling* que se propone en este documento.

Se sabe que estas técnicas se distinguen principalmente por los tipos de direcciones que ambas manejan, el *tunneling IPv6 over IPv4*, utiliza el tipo de direcciones *IPv6 con compatibilidad IPv4* y el *tunneling 6to4* utiliza una combinación de prefijo único del tipo **2002::/16** con una dirección IPv4 única globalmente.

Otro aspecto importante entre estas técnicas, es el hecho de que en el *tunneling 6to4* es necesario configurar un ruteador que utilice *Capa Dual IP* como punto de entrada del túnel, y el *tunneling IPv6-over-IPv4* el nodo que realiza el *tunneling* automático por ejemplo, debe ser configurado con direcciones *IPv6 con compatibilidad IPv4*.

Lo anterior da una idea de lo que se necesita para implementar el proyecto. Esto es, se requiere que el túnel que se diseñe satisfaga cuando menos las siguientes características:

- Ser compatible con varios tipos de direcciones, es decir no debe ser restrictivo en este sentido, aún más, en cuanto al prefijo de direcciones, este debe soportar idealmente cualquiera.
- La implementación del túnel deberá realizarla el programa que se diseñe, es decir, si algún nodo debe encapsular un paquete, esta operación deberá realizarla el mismo programa sin intervención del administrador, lo mismo deberá suceder si es necesario que el ruteador tenga que realizar algún tipo de configuración, esta deberá hacerse de manera similar a como se realiza en los nodos.
- El usuario deberá tener mínima intervención en cuanto a configuraciones se refiera durante el proceso de *tunneling* y de la manera en como este se realiza.
- El túnel que se diseñe deberá satisfacer las características comunes de ambos tipos de túneles que se describieron con anterioridad.
- Los nodos que implementen los túneles, deberán tener compatibilidad con la *Capa Dual IP*. Este último punto es necesario aunque no indispensable, debido a que es posible encapsular un paquete sin tener esta compatibilidad, aunque para las pruebas realizadas, todos los host son de *Capa Dual IP*.

Estos requisitos son los mínimos indispensables para realizar el proyecto. Existen aspectos, tales como la configuración manual del ruteador para enviar paquetes del nodo fuente al nodo destino en el caso de que el nodo fuente sea el encapsulador, pero estos se tratarán más adelante en el capítulo 4.

2.5 El futuro del *Tunneling* IPv6.

Se sabe que el *tunneling* es el encapsulamiento de tráfico IPv6 dentro de paquetes IPv4, de tal forma que estos puedan ser enviados sobre una red IPv4, permitiendo a las islas IPv6 y ruteadores comunicarse sin necesidad de actualizar la infraestructura que existe entre ellos.

El *tunneling* es una estrategia de desarrollo clave para ISP's y la empresa que desea conectarse al backbone (columna vertebral de la red) IPv6 durante el periodo de coexistencia entre IPv4 e IPv6.

El *tunneling* permite a los ISP's ofrecer un servicio IPv6 punto a punto sin necesidad de actualizar la infraestructura y sin impactar el funcionamiento de los servicios actuales de IPv4. También es posible interconectar islas IPv6 a través de la infraestructura existente IPv4.

Existen cuatro estrategias clave para el desarrollo de IPv6, éstas son:

1. Desarrollo de túneles IPv6 sobre IPv4. Estos túneles encapsulan el tráfico IPv6 dentro de paquetes IPv4 y son principalmente utilizados para comunicación entre redes aisladas IPv6 (islas IPv6) o para conexión remota a redes IPv6 sobre el backbone IPv4. Esta técnica incluye el uso de túneles manualmente configurados, túneles con encapsulamiento de ruteo genérico, tales como GRE¹ (Generic Routing Encapsulation) [HaS+94], mecanismos de túnel semiautomático y mecanismos de túnel completamente automático, tales como *IPv6 con compatibilidad IPv4 y 6to4*.
2. Desarrollo de enlaces de datos dedicados sobre IPv6. esta técnica habilita a dominios IPv6 aislados comunicarse utilizando la infraestructura de la capa 2 del modelo OSI (L2TP Layer Two Tunneling Protocol) para IPv4 pero utilizando circuitos virtuales permanentes (PVC Permanent Virtual Circuits) para Frame Relay o ATM.
3. Desarrollo de IPv6 sobre backbone MPLS (MultiProtocol Label Switching). Esta técnica permite a dominios aislados IPv6 comunicarse entre si a través del backbone MPLS IPv4. La transmisión esta basada en etiquetas más que en el encabezado IP mismo.
4. Desarrollo de IPv6 utilizando backbone con *Capa Dual IP*. Esta técnica permite a las aplicaciones IPv4 e IPv6 coexistir en un backbone de *Capa Dual IP*. Todos los ruteadores en la red necesitan ser actualizados a *Capa Dual IP* para lograr la comunicación IPv4 utilizando el stack IPv4 y la comunicación IPv6 utilizando el stack IPv6.

Múltiples técnicas están disponibles en diferentes puntos en la red, pero cada una requiere un pequeño cambio a la infraestructura del backbone o bien la reconfiguración de los ruteadores. En la tabla 2.1 se resumen los usos, beneficios y limitaciones de cada estrategia.

Además de las estrategias para desarrollar IPv6 dentro del ambiente IPv4, también son necesarios mecanismos o servidores de *Capa Dual IP* para permitir la comunicación entre aplicaciones que utilizan IPv4 y aplicaciones que utilizan IPv6, por ejemplo dispositivos que conecten navegadores *IPv6-only* a navegadores *IPv4-only*, o utilizando

¹ El GRE es un protocolo para realizar encapsulamiento de una capa de red arbitraria a otra capa de red arbitraria.

la *Capa Dual IP*, dispositivos que manejen servidores de correo *IPv4-only* y clientes de correo *IPv6-only*. Este tipo de mecanismos se van incrementando de manera importante a medida que IPv6 cambia de la fase de prueba a la fase de uso normal.

Estrategia de Desarrollo.	Usuario principal.	Beneficios.	Limitaciones.	Requerimientos.
Desarrollo de túneles IPv6 sobre IPv4.	El ISP ofreciendo un servicio inicial IPv6. La empresa esperando interconectar islas IPv6 a direcciones remotas IPv6.	Existe una demanda para IPv6 con un mínimo de inversión. Facilidad de implementar sobre la infraestructura existente IPv4.	Administración y diagnósticos complejos debido a independencia del túnel y topologías enlazadas.	Acceso a IPv4 a través de <i>Capa Dual IP</i> en un ruteador con direcciones IPv4 e IPv6 y acceso al DNS de IPv6.
Desarrollo de enlaces de datos dedicados sobre IPv6.	El ISP de la WAN desarrolla ATM o Frame Relay.	Proporciona un enlace punto a punto sin un gran impacto en el tráfico de la red.	Falta de aceleradores de hardware específicos para IPv6 y soporte para la administración de la red en el hardware actualmente desarrollado.	Acceso a la WAN a través de un ruteador <i>Capa Dual IP</i> con direcciones IPv4 e IPv6, y acceso al DNS IPv6.
Desarrollo de IPv6 sobre backbone MPLS.	ISP móviles ó ISP regionales desarrollando MPLS.	Integración de IPv6 sobre MPLS, así se puede trabajar con el hardware o software existentes.	Implementación requerida para trabajar con MPLS. Trae como consecuencia una gran sobrecarga de administración de la red.	Mínimos cambios en lado del cliente o en el lado del proveedor, dependiendo de la técnica.
Desarrollo de IPv6 utilizando backbone <i>Capa Dual IP</i> .	Pequeñas redes o negocios.	Facilidad de implementar en pequeñas redes con una mezcla de aplicaciones IPv4 e IPv6.	Administración compleja dual (IPv4 e IPv6) de los protocolos de ruteo. Mayor actualización para grandes redes.	Todos los ruteadores deben ser de <i>Capa Dual IP</i> con direcciones IPv4 e IPv6. Acceso al DNS IPv6. Suficiente memoria para las tablas de ruteo IPv4 e IPv6.

Tabla 2.1. Estrategias de desarrollo hacia IPv6.

Eventualmente, IPv6 se hace un protocolo de selección, puesto que los mecanismos permitirán heredar sistemas que son parte de IPv4, formar parte de la red IPv6. Estos mecanismos que trasladan entre protocolos IPv4 e IPv6, o en un servidor dedicado, o en un ruteador dentro de la red IPv6, deberán proveer un conjunto de herramientas para lograr el desarrollo incremental de IPv6 sin interrumpir ó alterar el tráfico IPv4.

Todas las estrategias de migración proveen un medio de enlace punto a punto entre IPv4 e IPv6. Esta conectividad requiere algún nivel de traslación entre los protocolos IPv4 e IPv6 ó entre el host y el ruteador ó entre host de *Capa Dual IP* con la aplicación a utilizar y el protocolo a usar.

Estos mecanismos de traducción se hacen más relevantes a medida que IPv6 se hace más relevante, y aún más, a medida que IPv6 se hace el protocolo de selección que va desplazando a IPv4 y formado parte de toda la red.

Otro factor importante a considerar se refiere al hecho de que algunas aplicaciones son más importantes que otras, por lo tanto la demanda de un tratamiento preferencial a través de la red LAN será necesaria.

Adicionalmente, las aplicaciones tienen diferentes demandas, tales como requerimientos en tiempo real de baja latencia y gran ancho de banda por mencionar algunas. Por tanto es deseable en una LAN manejar el concepto de QoS (Quality of Service) enfocado a aplicaciones con protocolos IPv4 e IPv6.

CAPÍTULO 3 Ruteo y Fragmentación.

3.1 Mecanismos de Ruteo.

Durante un enorme periodo de transición, deben coexistir sistemas basados en IPv6, sistemas basados en IPv4 y sistemas basados en IPv4-to-IPv6, entendiéndose esto último como una coexistencia entre ambos protocolos y el mecanismo para lograr la comunicación. En tal ambiente dual de protocolos interconectándose, ambas infraestructuras de ruteo IPv4 e IPv6 estarán presentes. Inicialmente es posible que dominios desarrollados con capacidad IPv6 puedan no estar interconectados globalmente vía IPv6 a la infraestructura de Internet, y por lo tanto necesitaran comunicarse sobre regiones de ruteo *IPv4-only*.

Con objeto de lograr tal ambiente mezclado, existe la necesidad de que sean los mecanismos de *tunneling* los que globalmente distribuyan la información de la capa de red IPv6 entre regiones IPv6 dispersas. Aún más, estas mismas técnicas pueden ser utilizadas en etapas posteriores de transición IPv4-to-IPv6 para rutear paquetes IPv4 en regiones aisladas *IPv4-only* sobre la infraestructura IPv6. Por lo tanto, es importante poner especial atención en los mecanismos de ruteo. Esto llevó a diseñar IPv6 con características mejoradas en el esquema de direccionamiento globalmente único y basado en prefijos más que en clases de direcciones. Esta característica mantiene las tablas de ruteo pequeñas logrando así un esquema eficiente de ruteo.

En una red internacional como lo es Internet, existen varios protocolos de ruteo utilizados. Aún más, la red está organizada como AS (Autonomous Systems), cada uno de los cuales es administrado por una entidad. Cada AS mantiene su propia tecnología de ruteo, la cual puede diferir entre AS's. El protocolo de ruteo utilizado dentro del AS es denominado IGP (Interior Gateway Protocol) y un protocolo distinto utilizado para transferir información de ruteo entre AS's es denominado EGP (Exterior Gateway Protocol), así RIP (Routing Information Protocol) fue diseñado para trabajar como un IGP de tamaño moderado [MaG+97] y utiliza una clase de algoritmo llamado de *Vector de Distancia* algunas veces conocido como *Algoritmo de Ford-Fulkerson*.

3.2 Ruteo en IPv4 e IPv6

La transición básica a IPv6 provee una transición de *Capa Dual IP*, comúnmente llamada *Capa Dual*, y los elementos de ruteo relacionados a esta transición incluyen los siguientes aspectos:

- Ruteo para paquetes IPv4.
- Ruteo para paquetes IPv6.
 - Paquetes IPv6 dentro de direcciones IPv6 nativas.
 - Paquetes IPv6 dentro de direcciones con compatibilidad IPv4.
- Operación de túneles manualmente configurados.
- Operación de encapsulamiento automático.
 - Localización de los mecanismos que encapsulan.
 - Confirmación exhaustiva de que el ruteo esta consistente con el encapsulamiento.

Los mecanismos requeridos para cumplir estos objetivos incluyen entre otros aspectos, los siguientes puntos:

- Cálculo de la ruta de la *Capa Dual IP (Dual Stack)*.
- Configuración manual de los túneles punto a punto.
- Filtrado de rutas para soportar *tunneling* automático.

Los mecanismos para el ruteo IPv4 e IPv6 involucran ruteo de la *Capa Dual IP*. Esto implica que las rutas son calculadas separadamente para direcciones IPv4 y para direcciones IPv6.

Así, por ejemplo, el uso de encapsulamiento automático, donde el punto de destino del túnel es determinado por la dirección empotrada en la dirección de destino *IPv6 con compatibilidad IPv4*, requiere consistencia de los ruteadores entre dominios de ruteo IPv4 e IPv6 para destinos que utilicen direcciones *IPv6 con compatibilidad IPv4*.

3.3 Ruteo.

El protocolo de información de ruteo **RIP** (Routing Information Protocol) es uno de los protocolos de gateway más ampliamente utilizados [HeC88]. **RIP** es un protocolo basado en el *algoritmo de vector de distancia*, fue diseñado para redes de tamaño moderado con algunas limitaciones:

- El protocolo es limitado a redes con trayectorias de cuando mas 15 saltos.
- Este protocolo depende de un proceso llamado *conteo infinito* para resolver ciertas situaciones, tales como los lazos de ruteo. Este proceso consume gran cantidad de ancho de banda antes de la solución de problema.
- También depende de medidas fijas para comparar rutas alternativas sin importarle parámetros de tiempo real tales como retraso, confiabilidad o carga. Además, estas métricas, pueden introducir inestabilidades² que el protocolo es incapaz de manejar.

² Para las inestabilidades producidas por la métrica utilizada, es necesario trabajar en los algoritmos de trayectoria de ruta para mejorar sus características.

RIPng (Routing Information Protocol Next Generation) es un protocolo basado en UDP (User Datagram Protocol). La trayectoria que un paquete toma a través de la red, es normalmente determinada por la red misma. Algunas veces, sin embargo la fuente requiere tener un control más preciso sobre la ruta que los paquetes tomen, aunque se tenga que sacrificar rapidez ganando seguridad.

El encabezado de ruteo en IPv6 permite que una trayectoria a través de la red sea predefinida, esto es agregar en el paquete IPv6 el campo opcional “*Routing Header*”. Este encabezado opera en forma similar a IPv4.

RIPng es un protocolo que permite a los ruteadores intercambiar información para trazar rutas a través de una red basada en IPv6. Cada ruteador que implementa RIPng debe tener una tabla de ruteo que tenga una entrada para cada destino alcanzable IPv6.

Cada entrada debe tener:

- El prefijo del destino IPv6.
- Una medida que indique el costo total de obtener un datagrama de un ruteador al destino.
- La dirección IPv6 del siguiente ruteador junto con la trayectoria hacia el destino, llamado el siguiente salto.
- Una bandera de cambio de ruta que indique si la información sobre la ruta ha cambiado recientemente.
- Varios contadores, tal que compartan cada determinado tiempo (aprox. 30 seg.) la información de la tabla de ruteo a los ruteadores vecinos.

La métrica RIPng de una red es un entero entre 1 y 15 inclusive. Esta métrica da el límite máximo de saltos en una trayectoria, el cual es 15. Es conveniente que el administrador de la red ponga una métrica con un valor adecuado para cada subred. Aún más, cada red deberá tener un prefijo de dirección destino y una longitud de prefijo asociado con la red.

Además, es posible también agregar rutas predefinidas para necesidades de tráfico particulares, este tipo de rutas es llamado *rutas estáticas*.

RIPng debe ser utilizado sólo en ruteadores, así cualquier ruteador que utilice RIPng, deberá tener a lo menos una o mas interfaces a cada subred.

Con objeto de que el protocolo provea información completa en el ruteo, cada ruteador en el AS debe participar en el protocolo. Si existen múltiples IGP's en uso en la red, debe existir a lo menos un ruteador el cual pueda proporcionar información de ruteo entre los protocolos.

Cada ruteador que utiliza RIPng tiene un proceso de ruteo que envía y recibe datagramas UDP en el puerto 521 (puerto de RIP), y todas las comunicaciones entre otros

ruteadores son en este mismo puerto. Consultas específicas pueden ser enviadas desde otros puertos diferentes al puerto RIPng, pero éstas deben ser dirigidas al puerto RIPng de la máquina destino.

El formato del paquete RIPng se ilustra en la figura 3.1, y el formato del mensaje que la entrada de la tabla de ruteo **RTE** (Route Table Entry) tiene se ilustra en la figura 3.2, así, cada mensaje contiene un encabezado RIPng, el cual consiste de un *comando* y un *número de versión*.

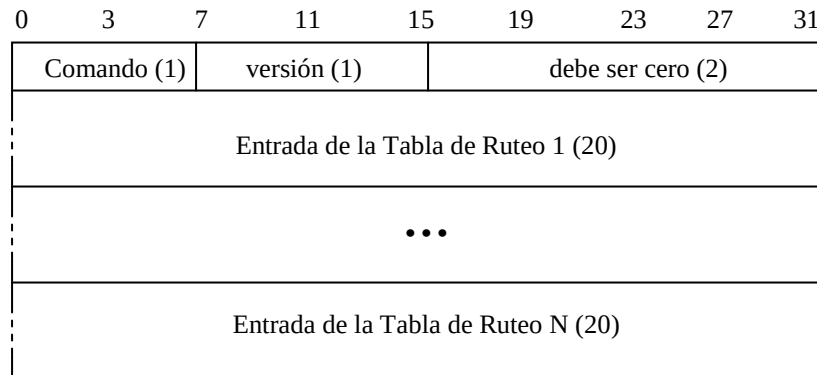


Figura 3.1. Formato de un paquete RIPng.

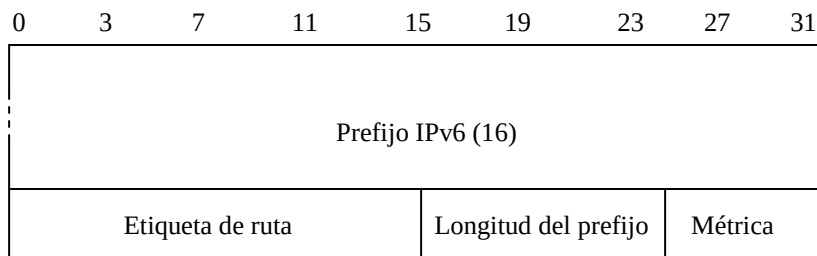


Figura 3.2. Formato de una entrada de la Tabla de Ruteo RTE.

Para cada uno de estos tipos de mensajes (ver figura 3.1), el resto del datagrama contiene una lista de los RTE's, de tal forma que cada RTE en la lista contiene el prefijo del destino, el número de bits significantes en el prefijo y el costo (métrica) de alcanzar el destino.

Los comandos implementados en RIPng versión 1 [MaG+97] son:

- **Solicitud.** Cuando se envía una solicitud, el ruteador remoto debe enviar toda o una parte de la tabla de ruteo.

- Respuesta. Es un mensaje que contiene toda o una parte de la tabla de ruteo del ruteador local. Este mensaje puede enviarse en respuesta a una solicitud, o puede ser una actualización no solicitada de una tabla de ruteo generada por el ruteador fuente.

El campo *Etiqueta de ruta* es un atributo asignado a la ruta la cual debe ser preservada y anunciada con la ruta, esto con el fin de proporcionar un método de separar ruteadores RIP *internos* (dentro de la misma subred) de RIP *externos* (en diferentes subredes).

El campo *Longitud Del Prefijo* es la longitud en bits de la parte significativa del prefijo (valor entre 0 y 128) iniciando del prefijo de más a la izquierda. Un prefijo con una longitud de prefijo de *cero* es designado una ruta *default*.

Se sugiere que el prefijo 0:0:0:0:0:0 sea utilizado cuando se especifica una ruta *default* y una ruta *default* es utilizada cuando no es conveniente listar cada posible red en la actualización RIPng.

El campo *Métrica* contiene un valor entre 0 y 15, especificando la métrica deseada para el destino, o bien, un valor de 16 indica que el destino es inalcanzable.

Una de las ventajas de utilizar RIPng, es que proporciona la capacidad de especificar la dirección del siguiente salto inmediato IPv6 a el cual el paquete deberá ser transmitido al destino especificado por una entrada en la tabla de ruteo RTE.

El propósito del siguiente salto RTE es para eliminar paquetes que están siendo ruteados a través de saltos extra en el sistema. Esta característica es útil cuando RIPng no se esta ejecutando en todos los ruteadores de la red.

La forma de actualizar la tabla de ruteo IPv6 es realizada mediante *contadores* ó *timers*, así, cada 30 segundos el proceso RIPng esta enviando mensajes no solicitados, conteniendo la tabla de ruteo completa a cada ruteador vecino.

Si existen varios ruteadores en la misma red, es necesario que se sincronicen, de tal forma que todas las listas de actualizaciones se realicen al mismo tiempo mediante mensajes *multicast*. Para realizar lo anterior se cuenta con dos *contadores*:

- *timeout*: Cuando el contador *timeout* expira, la trayectoria de ruta deja de ser valida, sin embargo esta es retenida en la tabla de ruteo por un corto tiempo hasta que los ruteadores vecinos puedan ser notificados que la trayectoria de ruta ya no es valida.
- *garbage-collection time*: Cuando el contador *garbage-collection time* expira, la trayectoria de ruta es finalmente eliminada de la tabla de ruteo.

Así de esta manera, el contador *timer* es inicializado cuando una ruta es establecida, y en cualquier momento, un mensaje de actualización de ruta es recibido del ruteador. Si

después de 180 segundos del último mensaje enviado, la trayectoria de ruta es considerada que ha expirado, por lo tanto deja de ser válida y el proceso de eliminación de esta trayectoria de ruta inicia.

La eliminación de una trayectoria de ruta ocurre por una de las siguientes dos razones: el contador *timer* espira o se recibió un mensaje de actualización de ruta del ruteador.

3.3.1 Procesamiento de solicitudes y respuestas.

Una solicitud es utilizada para preguntar por una parte o toda la tabla de ruteo del ruteador. Normalmente, las solicitudes son enviadas como datagramas *multicast* desde el puerto RIPng por los ruteadores que han obtenido una nueva entrada en la tabla de ruteo y están buscando completar rápidamente su tabla de ruteo.

Las solicitudes son procesadas entrada por entrada. Si no existen entradas, no hay una respuesta dada. Existe una diferencia en el manejo de la métrica para solicitudes de toda la tabla de ruteo o de una parte de esta. Si la solicitud es para toda la tabla de ruteo, un procesamiento normal es realizado, y si se requiere conocer solo una entrada específica, entonces, esta se busca en la tabla de ruteo y se envía como esta.

Una vez que la entrada ha sido validada, es necesario actualizar la métrica, agregando el costo de la red de la cual el mensaje llegó. Posteriormente, verificar que ya existe una ruta explícita para el prefijo del destino. Si no existe la ruta, tendrá que agregarse a la tabla de ruteo, lo cual consiste de:

- Poner el prefijo del destino y la longitud en el RTE.
- Poner la métrica al nuevo valor calculado.
- Poner la dirección del siguiente salto a la dirección del ruteador del cual el datagrama proviene ó la dirección del siguiente salto especificada por la RTE.
- Poner la bandera de cambio de ruta.
- Procesar la señal de salida para actualizarla.

Si existe una ruta, comparar la dirección del siguiente salto a la dirección del ruteador del cual el datagrama proviene. Si este datagrama es del mismo ruteador, reinicializar el *timeout*. Posteriormente, comparar las métricas. Si el datagrama es del mismo ruteador y la métrica es diferente que la que se tenía, hacer lo siguiente:

- Adoptar la ruta del datagrama. Esto es, cambiar el valor a la nueva métrica y ajustar la dirección del siguiente salto.
- Establecer la bandera de cambio de ruta.
- Si la nueva ruta es infinita, iniciar el proceso de eliminación, de otra forma reinicializar el *timeout*.

Una vez que las entradas han sido llenadas, se debe cambiar el comando de solicitud a respuesta, y enviar el datagrama de regreso al solicitante.

Una respuesta es recibida por alguna de las siguientes razones:

- Respuesta a una consulta específica.
- Actualización normal.
- Actualización disparada causada por un cambio hecho por el ruteador.

Debido a que el procesamiento de una repuesta puede actualizar la tabla de ruteo del ruteador, la repuesta debe ser verificada cuidadosamente para validarla, y esta debe ser ignorada si no proviene del puerto RIPng. Además, la dirección fuente del datagrama debe ser validada para verificar que el datagrama proviene de una dirección válida, es decir que es una dirección de enlace local. Aún más será necesario verificar que no se procesen copias del mismo datagrama enviado, es decir, de la misma dirección del ruteador fuente.

Una vez que el datagrama ha sido validado completamente, será necesario procesar las tablas RTE de respuesta una por una. Otra vez iniciando las siguientes pruebas de validación:

- ¿Es el prefijo de destino válido?
- ¿Es la longitud del prefijo válida?
- ¿Es la métrica válida?

Si cualquier chequeo falla, es necesario ignorar la entrada y proceder con la siguiente. Una vez que las entradas han sido validadas es necesario actualizar la métrica, agregando el costo de la red de la cual el mensaje llegó.

Ahora que se tienen las rutas especificadas, es necesario agregarlas a la tabla de ruteo, considerando los siguientes aspectos:

- La ruta ya existe en la RTE. En este caso es necesario comparar la dirección del siguiente salto a la dirección del ruteador de la cual el datagrama proviene. Si este datagrama es del mismo ruteador como está en la ruta, entonces se reinicializa el *timeout*, y se comparan las métricas. En caso de ser diferentes, se procede a tomar alguna de las siguientes acciones:
 - Si la nueva métrica es mayor a la métrica de la RTE, entonces se efectúa el proceso de eliminación de la nueva ruta.
 - Si la nueva métrica es menor a la métrica de la RTE, entonces se efectúa el proceso de adoptar la ruta del datagrama. Esto es, poner la nueva métrica en la RTE y ajustar la dirección del siguiente salto (sólo si es necesario). Además de poner la bandera de cambio de ruta para iniciar el proceso de actualización.
 - Si la nueva métrica es igual a la métrica de la RTE, entonces se deja la RTE sin cambio alguno, solamente se reinicializa el *timeout*. Aunque cabe aclarar que es necesario considerar el valor del *timeout*, puesto que si el valor actual

del timeout (antes de reinicializarlo) esta próximo a expirar, entonces será necesario cambiar el valor de la ruta a la nueva ruta e iniciar el proceso de actualización.

- La ruta no existe en la RTE. Entonces se inicia el proceso de agregar esta nueva ruta a la RTE (a menos que la métrica sea infinita, entonces no se efectúa acción alguna), el cual consiste de realizar las siguientes acciones:
 - Agregar el prefijo de destino y la longitud en la RTE.
 - Cambiar el valor de la métrica actual al valor de la nueva métrica.
 - Cambiar el valor de la dirección del siguiente salto al valor de la dirección del ruteador del cual el datagrama proviene o la dirección del siguiente salto especificada por el siguiente salto RTE.
 - Inicializar el timeout para la nueva ruta.
 - Activar la bandera de cambio de ruta.
 - Procesar una señal de salida para activar la actualización.

3.4 Ruteo en los túneles.

Un elemento útil en los mecanismos de ruteo, es poder contar con un protocolo de ruteo integrado que soporte el ruteo en ambos protocolos IPv4 e IPv6, pero hasta el momento de la escritura de este documento aún no se ha dado a conocer tal protocolo.

Existen varios métodos de *tunneling* (capítulo 2) que utilizan un ruteador como punto de entrada o salida del túnel, y dependiendo del tipo de dirección del nodo fuente, del nodo destino y la configuración de la red, es el tipo de túnel que se utiliza y la configuración del ruteador.

Antes de entrar en materia de ruteo, se precisará el concepto de *alcance*: cada dirección IPv6 tiene un *alcance* específico, esto es, una “*distancia*” topológica dentro de la cual la dirección puede ser usada como un identificador único para una interfaz, y el *alcance* de una dirección es codificado como parte de su dirección.

En términos de información de ruteo, la diferencia más importante entre IPv4 e IPv6, es el hecho de que IPv6 introduce el concepto de tres tipos de direcciones *unicast* (ver tabla 1.2 en Cap. 1) para las cuales define situaciones particulares cuando las direcciones de *alcance* deberán ser utilizadas, estos tipos son:

- Dirección de *alcance de enlace local*. Es utilizada únicamente para la identificación de interfaces dentro de un solo enlace.
- Dirección de *alcance de enlace de sitio local*. Es utilizada únicamente para la identificación de interfaces solamente dentro de un *sitio*, entendiéndose por *sitio* como una región de la topología que pertenece a una organización y es localizada dentro de una zona geográfica, algo similar a una LAN en IPv4.
- Dirección de *alcance global*. Es utilizada únicamente para la identificación de interfaces en cualquier parte de Internet.

Este diseño arquitectural de IPv6, provee soporte completo para el esquema de direccionamiento *unicast de alcance*. Así, una interfaz puede tener múltiples direcciones para comunicación dentro de diferentes *alcances*: de *enlace local*, de *sitio* o *global*. Es importante recordar que las direcciones de *enlace local* permiten comunicación local, aún cuando un ruteador no está presente. Algunos protocolos de ruteo requieren el uso de este tipo de direcciones de *enlace local*. Las direcciones de *enlace de sitio local* permiten que la comunicación esté administrativamente contenida únicamente dentro del sitio. Las conexiones de *enlace local* pueden sobrevivir a cambios en la información de prefijo global, es decir, al reenumerar el sitio.

IPv6 explícitamente asocia cada dirección con una interfaz. Los host *multihomed* pueden tener interfaces en más de un enlace ó en más de un sitio. Los enlaces y los sitios son internamente identificados usando *identificadores de zona*. Por tanto, el ruteo de tráfico *no global* y la selección de direcciones esta asegurado por la arquitectura de direcciones de *alcance* de IPv6.

3.5 Prioridades y Clase de Servicio.

Otro elemento fundamental en la transmisión de paquetes, se refiere al hecho de la necesidad de asignación de prioridades en la transmisión. IPv6 introduce el concepto de flujo, el cual es una serie de paquetes relacionados de una fuente a un destino que requieren un tipo particular de manejo por los ruteadores, por ejemplo el servicio en tiempo real. La forma en que se maneja este tipo de situaciones puede ser lograda agregando opciones a los datagramas, ésto es, utilizando las opciones *hop-by-hop* de IPv6 o por un protocolo separado, tal como el protocolo de reservación de recursos (*Resource Reservation Protocol RSVP*) (RFC 2205).

Los requerimientos de manejo para una etiqueta de flujo particular son conocidos como *información de estado*, esto es configurado en el ruteador. Cuando un paquete con una *flow label* (ver figura 1.2 en Cap. 1) conocida llega a un ruteador, éste puede decidir eficientemente como rutear y transmitir este paquete, sin tener que examinar el resto del encabezado del paquete.

Pueden haber múltiples flujos activos entre una fuente y un destino, también como tráfico que no esté asociado con cualquier flujo. Cada flujo es distintamente etiquetado por el campo *flow label* (ver figura 1.2 en Cap. 1) en el paquete IPv6 [PaC95].

El campo *flow label* es un generador de números aleatorios entre 1 y FFFFFFFF (hex), este es un número único cuando esta combinado con la dirección fuente. Cuando el campo *flow label* es cero, se dice que no existe *flow label* que este siendo utilizada.

Los mecanismos de flujo suponen que el estado asociado con una etiqueta de flujo dada está de alguna manera depositada en los ruteadores, de tal forma que los ruteadores saben como manejar los datagramas que llevan el campo *flow label*.

El campo *Traffic Class* (ver figura 1.2 en Cap. 1) permite a las aplicaciones especificar una cierta prioridad para el tráfico que ellas generan, introduciendo el concepto de *Class of Service* y reemplazando las funciones que fueron provistas en IPv4 por el campo *Type of Service* con objeto de permitir la diferenciación entre categorías del servicio de transferencia de paquetes. Los ruteadores basados en IPv4 normalmente tratan todo el tráfico igual, mientras que los ruteadores basados en IPv6 ahora deben actuar en los paquetes priorizados de la siguiente forma:

- Para las prioridades de 0 a 7, inician descartando estos paquetes cuando la red esta congestionada (congestión controlada).
- Para las prioridades de 8 a 15, estos intentan transmitir el paquete aún cuando la red esta congestionada descartando los paquetes de más baja prioridad. Las aplicaciones en tiempo real deberán optar por este rango de prioridades.

Hasta ahora se han analizado algunas de las funciones básicas provistas por el protocolo IP para lograr la comunicación entre dos o más redes físicas, utilizando los ruteadores como medio de enlace. Otro aspecto importante en el ruteo, es la fragmentación, este método de dividir los paquetes en partes más pequeñas y manejables es analizado en el siguiente punto.

3.6 Fragmentación.

El encabezado de fragmentación en IPv6 es utilizado por la fuente para enviar un paquete más grande que la MTU a su destino cuando no es posible enviar el paquete original debido a limitaciones en la red. La fragmentación en IPv6 es realizada por el nodo fuente (a diferencia de IPv4, donde la fragmentación es realizada por el ruteador que transmite el paquete a su destino). Esta técnica es utilizada para enviar un paquete mayor que la MTU a su destino, dividiendo el paquete en fragmentos menores que la MTU y enviando cada fragmento como un paquete separado para que sea reensamblado en el nodo destino.

Para cada paquete que es transmitido de manera fragmentada, el nodo fuente genera un valor de identificación que debe ser diferente que el de cualquier otro paquete fragmentado enviado recientemente por la misma fuente con la misma dirección fuente y destino. La longitud de los fragmentos debe ser seleccionada tal que el paquete fragmentado resultante sea menor que la MTU.

Un paquete que requiere fragmentación se considera que consiste de dos partes (ver figura 3.3a y 3.3b):

- La parte fragmentable. Es el balance del paquete, el cual puede incluir cualquier extensión de encabezado que será procesado al final del nodo destino por las capas superiores. Los fragmentos aquí divididos deberán ser múltiplos de 8 bytes, excepto el último fragmento, el cual puede ser de tamaño arbitrario no excediendo la MTU. Cada fragmento aquí consiste de tres partes establecidas en el nuevo encabezado:

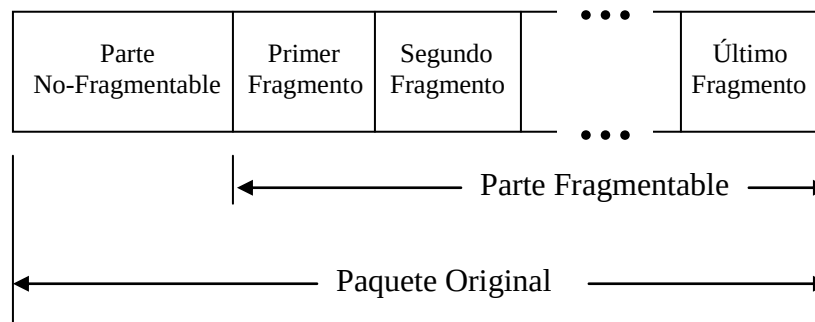


Figura 3.3a. Fragmentación de un paquete que excede el MTU

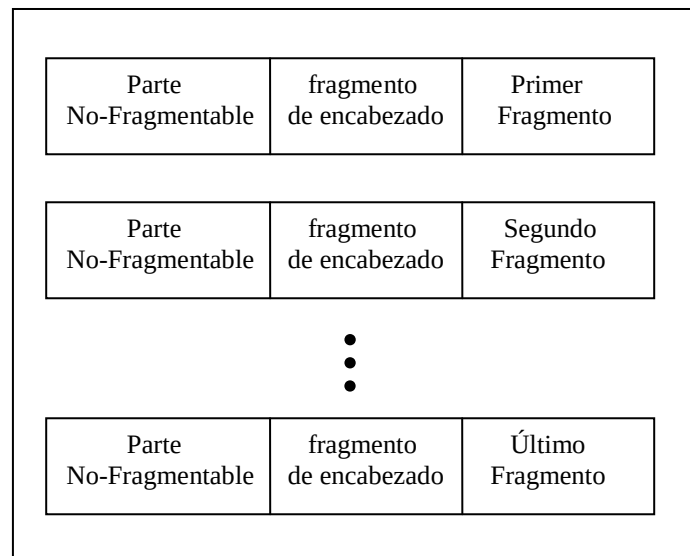


Figura 3.3b. Paquete fragmentado con un tamaño mayor a la MTU.

- Parte no fragmentable. Contiene el campo *Payload Length* modificado, que asocia el nuevo tamaño del paquete.
- Encabezado del fragmento. Aquí se le agrega el encabezado extra al paquete Next Header, indicando que paquetes adicionales vienen en camino.
- Datos del fragmento. Son un fragmento de los datos que serán enviados.

- La parte no fragmentable. Esta incluye un encabezado IPv6 más cualquier extensión de encabezado que deberá ser procesado en la ruta al destino.

En el reensamble, los paquetes fragmentados que tienen la misma dirección fuente, destino y número identificador del paquete son pegados de acuerdo a las siguientes reglas.

La parte *no fragmentable* de un paquete reensamblado consiste de todos los encabezados, pero sin incluir el encabezado de fragmento del primer paquete fragmentado con los siguientes dos cambios:

- El campo **Next Header** del ultimo encabezado de la parte *no fragmentable* es obtenido del campo **Next Header** del primer fragmento del encabezado.
- La longitud de la carga del paquete reensamblado es calculada de la longitud de la parte *no fragmentable* y la longitud del desplazamiento del último fragmento.

La parte fragmentable del paquete reensamblado es construida de los fragmentos siguientes al encabezado del fragmento en cada uno de los paquetes. La longitud de cada fragmento es calculada de la resta de la longitud de la carga del paquete, la longitud de los encabezados del paquete y el fragmento mismo. El paquete final de los paquetes fragmentados, no lleva un encabezado fragmento.

Es posible que existan algunas condiciones de error en el reensamble de paquetes fragmentados, estas condiciones son:

- Fragmentos insuficientes son recibidos para completar el reensamble de un paquete dentro de los últimos 60 segundos de recepción del primer paquete fragmentado. Si esto sucede, todos los fragmentos recibidos de ese paquete fragmentado, deben ser desechados.
- Si la longitud de un fragmento derivado de un paquete no es múltiplo de 8, a excepción del último fragmento del paquete, este debe ser descartado y un mensaje ICMP de “Problema de Parámetros” sea enviado.
- Si la longitud del desplazamiento del fragmento es tal que la longitud del fragmento excede los 65,535 bytes, este debe ser descartado y un mensaje ICMP de “Problema de Parámetros” debe ser enviado.

Aunados a estos errores, es posible que surjan otros errores, los cuales se tratan en el siguiente punto.

3.7 Manejo de errores.

Cuando se envían paquetes dentro de un túnel, es posible que en algunas ocasiones se produzcan mensajes de error ICMP de los ruteadores dentro del túnel. Estos paquetes son direccionados al nodo encapsulante, debido a que esta es la fuente IPv4 del paquete encapsulado.

Los mensajes de error ICMP “*packet too big*” son manejados de acuerdo al descubrimiento de la trayectoria MTU IPv4 y la trayectoria resultante MTU es registrada en la capa IPv4. La trayectoria MTU registrada es utilizada por IPv6 para determinar si un error ICMP “*packet too big*” ha sido generado.

El manejo de otros tipos de mensajes de error ICMP depende de cuanta información ha sido incluida en el campo “***Packet In Error***”, el cual mantiene el paquete encapsulado que causó el error.

En la figura 3.4 es muestra los mensajes de error.

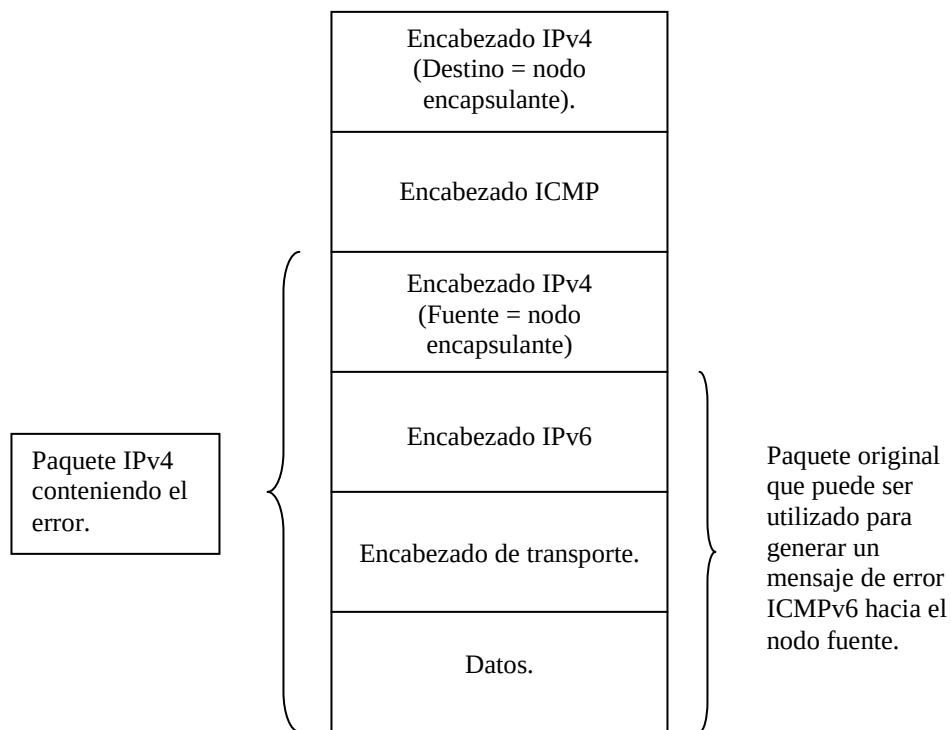


Figura 3.4. Mensaje de error regresado al nodo encapsulante.

3.7.1 Mensajes ICMPv6.

ICMP es considerado una parte integral de la capa IP y debe ser implementado en los módulos IP contenidos en ambos host y ruteadores. Los mensajes ICMP están contenidos dentro de datagramas IP, con el encabezado IP precediendo al mensaje ICMP y los datos.

ICMPv6 es la nueva versión de ICMP, la cual está diseñada para IPv6. ICMPv6 incluye funciones que están definidas por el protocolo MLD, y al igual que ICMP, este debe ser implementado por los host y ruteadores.

ICMPv6 es utilizado para reportar errores en el procesamiento de paquetes, comunicaciones en redes internas y reporte de miembros *multicast*.

El proceso de descubrir a los vecinos permite a los nodos en el mismo enlace descubrir la presencia de otros nodos, determinar las direcciones respectivas del nodo y encontrar ruteadores para una trayectoria hacia otras redes.

Algunos de los mensajes ICMPv6 son, permitir al proceso de descubrimiento de vecinos que contenga prefijos de sitio para lograr un enlace entre host y así disminuir la carga de red, otra es la de preguntar a un nodo IPv6 para que provea un nombre FQDN (Fully Qualified Domain Name), otras más son el descubrimiento de la MTU hacia la trayectoria destino para optimizar la trayectoria de comunicación.

Como tal, los mensajes ICMPv6 son precedidos por un encabezado IPv6 y cero o más extensiones de encabezado. Estos mensajes tienen tres campos que son comunes a todos, los mensajes más un cuerpo de mensaje de longitud variable cuyo contenido depende del tipo de mensaje que esta siendo transmitido. En la figura 3.5 se muestran algunos de los mensajes ICMPv6.

Cada formato de mensaje de error es similar en sus primeros 32 bits, esto es contienen tres campos como se aprecia en la figura 3.5, además de una copia parcial (o total) del paquete original transmitido.

Los nodos IPv6 que descartan un paquete pueden enviar un mensaje de error ICMPv6 al destino original. Existen cuatro casos en los cuales puede ocurrir un error y cada una de estas circunstancias produce un mensaje de error diferente, estos son:

- *El destino no fue alcanzado.* Este mensaje puede tener uno de los siguientes valores:
 - Código 0. No existe ruta hacia el destino. Puede deberse a que no existe una ruta en la tabla de ruteo y el ruteador no conoce o no puede aplicar la trayectoria por defecto al destino.
 - Código 1. El enlace con el destino está administrativamente prohibido. Este mensaje puede ser obtenido debido a un filtro firewall que prohíbe la transmisión del paquete.
 - Código 2. No asignado. Inicialmente fue asignado al mensaje “Not neighbord”, ahora no tiene mensaje.
 - Código 3. Dirección inalcanzable. Este mensaje es debido a una falla en el envío causada por una dirección de la capa de enlace no resuelta.
 - Código 4. Puerto inalcanzable. Este mensaje es utilizado cuando la capa de transporte del nodo destino no tiene un proceso escuchante activo.

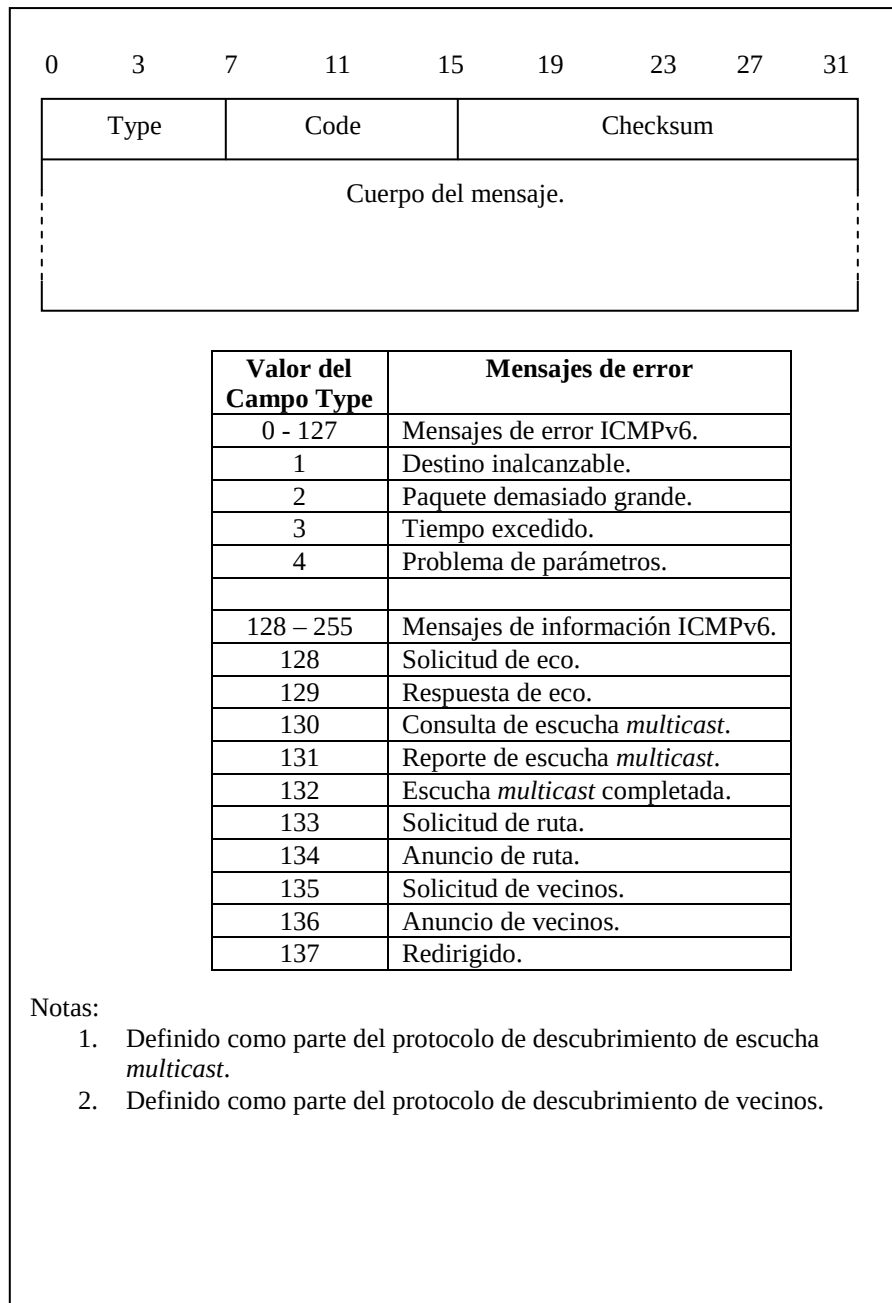


Figura 3.5. Distintos mensajes ICMPv6.

- *El paquete es demasiado grande.* Este mensaje es utilizado en conjunto con el proceso de descubrimiento de la MTU. En este proceso, el host envía un paquete con el máximo tamaño posible, para ver si este puede enviarse a su destino, si no, un mensaje “*Paquete demasiado grande*” es obtenido junto con la MTU más pequeña que el ruteador soporta, posteriormente, un segundo intento es realizado

y se retorna el mismo mensaje de error junto con la MTU en caso de falla en la entrega del paquete o de lo contrario la entrega es exitosa. El mismo procedimiento se repite hasta que la entrega sea exitosa. También es posible obtener el mismo mensaje en respuesta a un paquete recibido con una dirección de destino *multicast*, una dirección *broadcast* ó *multicast* de la capa de enlace.

- *El paquete excedió el tiempo de vida.* Este mensaje puede tomar uno de los siguientes valores:
 - Código 0. Excedido el campo hop limit. En este caso el paquete se desecha y probablemente exista un lazo de ruteo.
 - Código 1. Excedido el tiempo de reensamble del paquete. Esto sucede comúnmente cuando se extravía un fragmento.
- *Problema de parámetros.* Este error es encontrado en todos los casos cuando un campo del encabezado del paquete no puede ser procesado. Cuando esto ocurre, el paquete es descartado y este mensaje es obtenido indicando el tipo y localización del problema. El nodo que recibe este tipo de errores debe notificarlo a los protocolos de la capa superior. Este error puede tomar cualquiera de los siguientes valores:
 - Código 0. Encontrado un campo del encabezado erróneo.
 - Código 1. Encontrado un tipo de Encabezado Siguiente desconocido.
 - Código 2. Encontrada una opción IPv6 desconocida.

Los Mensajes de Solicitud/Respuesta de *eco* son enviados también como mensajes ICMPv6 y son utilizados para regresar una respuesta a la solicitud de *eco*. Todos los nodos deben implementar una interfaz en la capa de aplicación (capa 7 del modelo OSI) que facilite el uso de mensajes ICMPv6.

La dirección fuente de un mensaje de respuesta de *eco* enviado a un mensaje de solicitud de *eco* a través de una dirección *unicast* debe ser la misma como la dirección de destino del mensaje de solicitud de *eco*.

Los mensajes de respuesta de *eco* deben ser pasados a la interfaz de usuario ICMPv6, a menos que el mensaje de respuesta de *eco* correspondiente sea originado en la capa IP.

Existen otros múltiples mensajes que no serán tratados aquí, los cuales van más allá del alcance de este documento.

CAPÍTULO 4 Implementación.

4.1 Introducción.

Los protocolos de comunicación son parte fundamental de la conexión entre las computadoras. Todo enlace entre computadoras debe utilizar algún protocolo. Es bien sabido que Internet es una red multiprotocolos. Este proyecto se dirigió hacia un protocolo en particular, el IPv6. Este protocolo de comunicación esta aún en una fase de desarrollo y existe bastante trabajo por realizar a futuro. Además dicho protocolo todavía no es un estándar y en parte se debe a que se encuentran en desarrollo características importantes del protocolo, tales como la seguridad, encriptación, movilidad, etc. Esto hace que todavía no se tenga la confianza necesaria y suficiente para implementar IPv6 en una organización ó industria que dependa completamente de Internet. Es cierto, que ha habido grandes avances en materia, pero todavía queda bastante por hacer.

Es probable que pasen muchos años hasta que se empiece a implementar de manera confiable este protocolo en la industria, mientras tanto quienes estén interesados es coadyuvar en el desarrollo de IPv6 son bienvenidos.

La transición de IPv4 a IPv6 debe satisfacer algunas necesidades separadas. A corto plazo, es necesario definir las tecnologías y métodos específicos de transición en redes IPv4 hacia redes IPv6. A largo plazo, se tendrán que lograr estrategias de planeación operacional en redes basadas en banda ancha (principalmente), desarrollando métodos los cuales permitan guiar estas estrategias, entendiendo que deberá existir una enorme interacción por un largo periodo de coexistencia entre ambos protocolos. Además será necesario garantizar que el Internet del futuro (IPv6) sea rápido, seguro, confiable y fácil de administrar entre otras características.

Este proyecto provee una alternativa para resolver uno de los problemas de la transición de IPv4 hacia IPv6, esto es, cuando se tiene que atravesar una red IPv4 (redes comunes en la actualidad) y se desea efectuar un enlace entre dos nodos utilizando el protocolo IPv6, es necesario agregar un encabezado extra al paquete enviado, para que sea compatible con los protocolos que se manejan en la red (IPv4), de tal manera de lograr encapsular el paquete original dentro de otro protocolo [LoM+03].

Existen algunos métodos para lograr una conexión IPv6 a través de la infraestructura de ruteo IPv4, entre ellos se encuentran los túneles manualmente configurados, túneles automáticos, túneles *6to4*, NAT (Network Address Translator) entre otros, los cuales modifican el encabezado del paquete original, lo envían y después lo convierten al paquete original.

Algunas de estas técnicas requieren compatibilidad con la *Capa Dual IP* entre cada uno de los puntos de entrada y salida del túnel, esto es, soportar ambos protocolos IPv4 e IPv6 en la capa de *red* del modelo de referencia OSI.

La herramienta aquí presentada, provee un enlace entre nodos IPv6 los cuales atraviesan por una infraestructura de red IPv4, de tal forma que el usuario final pueda lograr un enlace entre los nodos IPv6 sin necesidad de tener que realizar algún tipo de configuración en el sistema operativo. Es importante notar que este tipo de herramientas permiten lograr el enlace entre host IPv6 de manera rápida y simple. Otra ventaja de utilizar este tipo de herramientas, es para llevar a cabo pruebas utilizando el protocolo IPv6 y con los resultados obtenidos planear una estrategia de migración hacia el Internet de la próxima generación (IPng ó IPv6).

Este trabajo esta enfocado en el método *tunneling* el cual agrega un encabezado extra a paquete original, lo envía y se lo quita posteriormente, ya sea en el destino o en algún ruteador (punto de salida del túnel). El túnel es una trayectoria de transmisión entre dos nodos, en los cuales la carga del paquete es el paquete original.

4.2 Configuración e instalación.

La infraestructura que permitirá culminar con éxito este proyecto, depende en parte de las herramientas disponibles, las cuales están al alcance del proyecto. Aunado a lo anterior, otro factor importante, son las fuentes de información existentes, tales como sitios Web, libros y RFC's, Draft's, entre otros.

Al inicio del proyecto, fue necesario contar con algunas herramientas indispensables, entre ellas se observo la necesidad de contar con un sistema operativo flexible y con capacidad para hacerle configuraciones y modificaciones necesarias para lograr los objetivos deseados, otra es el lenguaje de programación a utilizar.

Se opto por seleccionar el SO (Sistema Operativo) *Linux SUSE versión 7.2* el cual es un SO caracterizado por su estabilidad y buen desempeño, con un entorno grafico amigable.

Se seleccionó el *Lenguaje de Programación C*, puesto que este lenguaje ofrece un control total de los sockets en cualquier capa del modelo de referencia TCP/IP. Además, la mayoría de las aplicaciones que utiliza Linux, están escritas en este lenguaje.

Una vez decidido el lenguaje de programación, es necesario definir el tipo de sockets a utilizar, puesto que de ello depende el nivel de fiabilidad y rapidez. Para ello, se realiza una rápida comparación de las características del tipo de paquetes y de los tipos de sockets y en base a ello se selecciona el tipo de socket que satisfaga las expectativas. En la tabla 4.1 se presenta una descripción de las principales características de los tipos de paquetes.

	Raw	ICMP	UDP	TCP
Sobrecarga fija (bytes).	20-60	20-60 + [4]	20-60 + [8]	20-60 + [20-60]
Tamaño del mensaje (bytes).	65,535	65,535	65,535	Alta
Fiabilidad.	Baja	Baja	Baja	Alta
Tipo de mensaje.	Datagrama	Datagrama	Datagrama	Flujo
Rendimiento.	Alto	Alto	Medio	Bajo
Integridad de los datos.	Baja	Baja	Media	Alta
Fragmentación.	Si	Si	Si	Baja

Tabla 4.1. Características de los tipos de paquetes.

En la tabla 4.2 se presenta la relación entre los tipos de sockets y los paquetes, (utilizando el modelo TCP/IP) para acceder a todas las capas de la pila IP y usando la llamada a sistema **socket()**.

Modelo TCP/IP	Acceso de Usuario/Programador
4 – Aplicación.	TELNET, FTP, IRC, Navegador, etc.
3 – De Host a Host (TCP).	socket(PF_INET, SOCK_STREAM, 0);
3 – De Host a Host (UDP).	socket(PF_INET, SOCK_DGRAM, 0);
2 – Internetwork (ICMP).	socket(PF_INET, SOCK_RAW, IPPROTO_ICMP);
2 – Internetwork (IP).	socket(PF_INET, SOCK_RAW, “tipo de protocolo”);
1 – De acceso a la red.	socket(PF_INET, SOCK_PACKET, “filtro”);

Tabla 4.2. Relación entre las capas TCP/IP y los tipos de socket's.

De la tabla 4.2, se puede deducir que para lograr un control total del paquete en la capa de *Internet* del modelo TCP/IP, que es donde trabaja el *túnel*, se debe utilizar el socket tipo **SOCK_RAW**. Con el **SOCK_PACKET**, Linux permite accesos de lectura a los mensajes del controlador de bajo nivel, y así mismo, se tiene acceso a todos los mensajes de la red.

Otros factores a definir son el tipo de túnel a utilizar, el tipo de direcciones a configurar, tales como direcciones unicast de enlace local global o de sitio en IPv6, además del tipo de subred a diseñar y su implementación, así mismo como del protocolo a utilizar en el ruteo entre otros factores.

Un aspecto importante a considerar es la asignación de direcciones IPv4 e IPv6. Al asignar una dirección IPv6, es necesario determinar que tipo de dirección se tiene que asignar. Se sabe que en IPv6 existen muchos tipos de direcciones disponibles, entonces se debe estar seguro de elegir el tipo de dirección adecuada a las necesidades del proyecto.

El tipo de direcciones IPv4 a utilizar son direcciones *Clase C*, es decir en el rango de 192.168.X.X, aunque es posible utilizar cualquier tipo de direcciones clase A, B, ó C. Estas direcciones son adecuadas para la subred que se diseño y que se explica en detalle en la sección 4.2.1.

En la tabla 1.2 (Capítulo 1) se muestra la estructura de direcciones IPv6. De esta tabla se observa que la estructura de direcciones adecuada es la dirección *unicast de enlace local*. Este tipo de direcciones es la adecuada para realizar los enlaces entre nodos en una subred, y si se requiere realizar un enlace entre dos redes separadas atravesando Internet, es necesario elegir el tipo de dirección *unicast global agregable*.

Nótese que no quiere decir que éstos sean los únicos tipos de direcciones, sino que para el proyecto realizado, estas direcciones son las adecuadas para lograr conectividad entre nodos, aunque también se podrían utilizar direcciones *anycast* ó *multicast*, pero esto esta fuera del alcance de este proyecto.

La dirección *unicast* de enlace local tiene el rango de direcciones “FE80 :: / 10”, con una longitud de máscara de 10 bits y como prefijo “1111 1110 10”. Así la asignación de direcciones de los nodos y del ruteador se basará en este rango.

Una vez que las herramientas han sido definidas, tales como el Sistema Operativo, el lenguaje de programación y el tipo de sockets, el siguiente paso es diseñar la subred que nos servirá para realizar pruebas.

Para lograr este propósito, se procede a analizar cuales son los requerimientos indispensables de conectividad que debe satisfacer la implementación del túnel:

1. Es necesario realizar un enlace en el mismo equipo. Esto es posible lograrlo enviando un paquete en una terminal a la dirección *loopback* y recuperándolo en otra terminal del mismo nodo, comprobando que realmente se satisface el enlace.
2. Es necesario realizar un enlace entre dos nodos en la misma subred. Esto es posible enviando un paquete a la dirección del nodo destino y comprobando que efectivamente se realiza el encapsulamiento.
3. Es necesario realizar un enlace entre un nodo de la subred y el ruteador. Esto se realiza enviando un paquete al ruteador y comprobando que efectivamente se realiza la función de encapsulado.
4. Es necesario realizar un enlace entre dos nodos en diferente subred. De manera similar, aquí es necesario enviar un paquete al nodo destino con la característica que debe atravesar el ruteador y así mismo el ruteador debe enviar el paquete al destino correspondiente.

Estos pasos a seguir en el procedimiento de túnel es necesario llevarlos en ese orden para comprobar que efectivamente se satisfacen los requisitos del encapsulamiento.

Una vez comprendidos los requerimientos básicos y requisitos del encapsulamiento, se procedió a diseñar la subred que nos servirá como plataforma de pruebas.

4.2.1 Especificaciones en el diseño de la subred.

El diseño de la subred está basado en el trabajo que se quiere realizar. Este diseño no requiere características especiales de planeación. La subred que se utilizara en el proyecto basta con que este *segmentada*.

Para las pruebas realizadas es suficiente con tener un nodo en cada subred y el ruteador configurado para aceptar las peticiones, este último es el enlace entre las subredes.

El ruteador que se utilizó para el proyecto, fue una máquina con el SO Linux *SuSE* con dos tarjetas de red y configurada de tal manera que paquetes provenientes de una interface fueran capaces de salir por la otra interface hacia la otra subred utilizando el protocolo IGP (Interior Gateway Protocol). Se decidió utilizar este protocolo, debido al hecho de que se configuraron rutas estáticas en esta máquina y aún más, éstas pueden manejarse adecuadamente mediante un programa utilizando funciones tales como `rt_NRupdate()`. A continuación se ilustra en la figura 4.1 el esquema de la subred.

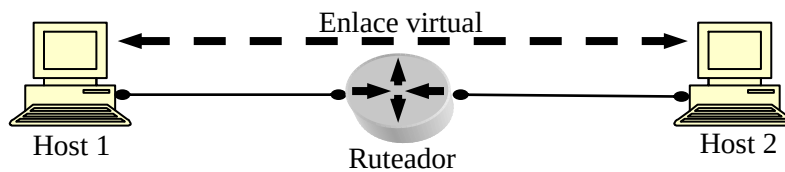


Figura 4.1 Subred utilizada para las pruebas.

Nótese que este diseño simple se puede extender a una subred más grande sin necesidad de configuraciones adicionales en los host y una mínima configuración adicional en el ruteador, esto es, se puede añadir un hub o switch en cada extremo, e incluso un nodo puede atravesar más ruteadores o gateways, basta con que se modifique la tabla de ruteo en el ruteador (en caso de ser rutas estáticas).

Una vez que se tiene la subred en la que se trabajará, es conveniente realizar primero pruebas utilizando solo un protocolo a la vez y posteriormente manejar ambos protocolos para realizar el túnel. Esto es con el fin de asegurar que los enlaces establecidos satisfacen las necesidades requeridas de comunicación y confiabilidad, además de verificar que el ruteo se realice de la manera predeterminada.

Para lograr este propósito, se diseñaron programas utilizando sockets que realizan la comunicación utilizando solamente el protocolo IPv4, así mismo, se realizaron programas

que realizan la comunicación utilizando el protocolo IPv6 con sockets. Esto se describe a continuación en el punto 4.3.

4.3 Pruebas preliminares.

Se realizaron algunas pruebas iniciales antes del proceder a realizar el encapsulamiento. El objetivo de este punto es determinar satisfactoriamente cual es el comportamiento de los paquetes que viajan a través de la subred. A continuación se detallan los enlaces preliminares realizados.

Se realizó un programa que envía un paquete en la misma subred ya sea al mismo host o a otro host en la misma subred. Aunque sólo se tiene un nodo en cada subred, se tiene la seguridad de que efectivamente es posible enviar un paquete a dos nodos de la misma subred. Para comprobar lo anterior, bastó con agregar un nodo extra en la subred, conectado mediante un *hub* 3Com de 100 Mhz de 16 puertos como se ilustra en la figura 4.2 (cualquier *hub* genérico de esta velocidad puede lograr el enlace).

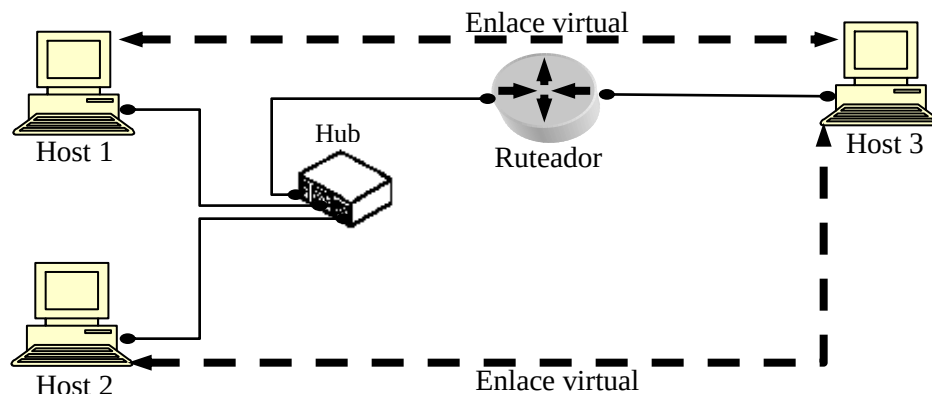


Figura 4.2. Subred utilizada para verificar el enlace en la misma subred.

Para enviar un paquete a otro host en otra subred basta con configurar el ruteador para que encamine los paquetes a la otra subred, así que efectivamente se cumplen las condiciones mínimas aceptables para enviar paquetes entre subredes utilizando el protocolo IPv4 con los programas desarrollados.

De manera similar se realizó un programa que envía paquetes en la misma subred ya sea en el mismo host o en otros host de la misma subred. Y para enviar paquetes a otra

subred, fue necesario configurar el ruteador para que acepte paquetes IPv6³ y los encamine a otras subredes. Por lo tanto se observó de manera similar a lo sucedió con el protocolo IPv4 que se cumplieron las mínimas condiciones de conectividad entre host IPv6.

También se realizaron pruebas en las cuales se transmitieron paquetes utilizando ambos protocolos. Para lograr lo anterior, se enviaron paquetes dentro de la misma subred y hacia la otra subred utilizando los programas diseñados para el protocolo IPv4 y los programas diseñados para el protocolo IPv6. En este punto, el envío de paquetes se realizó ejecutando los programas diseñados para cada protocolo simultáneamente. Mediante estas pruebas, se pudo comprobar una de las más importantes características de Internet, la de ser multiprotocolos.

Cabe aclarar en estas pruebas preliminares, se utilizaron sockets del tipo *sock_stream* y *sock_dgram* para cada uno de los enlaces. Estos sockets son orientados a conexión y sin conexión respectivamente.

Estos programas que se diseñaron tuvieron el fin de comprobar el enlace entre nodos. Las pruebas preliminares realizadas fueron las siguientes:

- Se realizaron conexiones entre nodos IPv4 utilizando enlaces orientados a conexión (*sock_stream*).
- Se realizaron conexiones entre nodos IPv4 utilizando enlaces sin conexión (*sock_dgram*).
- Se realizaron conexiones entre nodos IPv6 utilizando enlaces orientados a conexión (*sock_stream*).
- Se realizaron conexiones entre nodos IPv6 utilizando enlaces sin conexión (*sock_dgram*).
- Se realizaron conexiones entre nodos IPv4 e IPv6 de manera simultánea utilizando enlaces orientados a conexión (*sock_stream*).
- Se realizaron conexiones entre nodos IPv4 e IPv6 de manera simultánea utilizando enlaces sin conexión (*sock_dgram*).

Una vez logrado el primer objetivo, de realizar un enlace entre nodos, el siguiente paso es construir el encabezado y anexarle la información deseada.

4.4 Implementación.

Se iniciará este punto definiendo adecuadamente cuales son las direcciones a manejar en el túnel de entrada y en el túnel de salida. Esto dependerá del nodo encapsulador, esto es, si el nodo encapsulante es el mismo nodo fuente, entonces el paquete enviado deberá tener como dirección de entrada del túnel a la dirección del mismo nodo encapsulador, y

³ El procedimiento de configuración realizado se describe en el apéndice D.

si el nodo encapsulante es distinto del nodo fuente, entonces la dirección de entrada del túnel deberá ser la dirección del nodo encapsulante.

Algo similar deberá ocurrir al asignar la dirección de destino. Si el nodo desencapsulante es el mismo que el nodo destino, entonces la dirección correspondiente a la dirección de salida del túnel deberá ser la dirección del mismo nodo desencapsulador. Y si la dirección el nodo desencapsulante es distinta al nodo destino, entonces la dirección de salida del túnel deberá corresponder a la dirección del nodo desencapsulador.

El proceso de la asignación de direcciones se debe realizar de acuerdo a la tabla 4.3, y esta basada en el hecho de que el nodo encapsulante es el que determina la dirección de entrada del túnel, así mismo el nodo desencapsulador es el que determina la dirección de salida del túnel.

Nodos	Dirección fuente del paquete.	Dirección de entrada del túnel.	Dirección de salida del túnel.	Dirección destino del paquete.
Nodo emisor = nodo encapsulador.	Dirección IPv6 del nodo fuente.	Dirección IPv4 del nodo fuente.		
Nodo emisor $\neq$ nodo desencapsulador.	Dirección IPv6 del nodo fuente.	Dirección IPv4 del nodo encapsulador.		
Nodo receptor = nodo desencapsulador.			Dirección IPv4 del nodo destino.	Dirección IPv6 del nodo destino.
Nodo receptor $\neq$ nodo desencapsulador.			Dirección IPv4 del nodo desencapsulador.	Dirección IPv6 del nodo destino.

Tabla 4.3. Formato de asignación de direcciones.

En base a la tabla anterior, se puede determinar con facilidad cual es la dirección correspondiente en cada punto del proceso de *tunneling*, teniendo en consideración que las direcciones IPv4 son direcciones *clase B* y las direcciones IPv6 son direcciones del tipo *unicast global agregable*.

Definidas las direcciones, se procede a iniciar la construcción del encabezado y agregarle la información, ésta será el paquete IPv6 enviado, ya sea que la fuente encapsule el paquete, o sea otro nodo el encapsulador.

Durante el proceso de construcción del encabezado IPv4 que se anexará, es necesario definir algunos campos del mismo, para ello, se considera para propósito de simplificar el

método de *tunneling*, que no se realizará fragmentación, este punto se menciona más adelante.

Así, los campos del encabezado IPv4 tendrán que tomar los siguientes valores:

- *version* → 4
- *IHL* → 5
- *Type of Service* → Este valor es copiado del campo *Traffic Class*. De acuerdo a [Bla+98], la semántica de estos campos en IPv4 e IPv6 es idéntica.
- *Total Length* → Es el valor del campo *Payload Length* del encabezado IPv6 más el tamaño del encabezado IPv4.
- *Identifier* → 0
- *Flags* → la bandera *DF* (Don't fragment) = 1
la bandera *MF* (More fragments) = 0 Como era de esperarse.
- *Fragment Offset* → 0
- *Time To Live* → Este valor es copiado del campo *Hop Limit* en IPv6⁴.
- *Protocol* → Este valor es copiado del campo *Next Header* de IPv6
- *Header Checksum* → Este valor es calculado en el proceso de creación del encabezado IPv4.
- *Source IP* → Esta es obtenida en base a la tabla 4.3.
- *Destination IP* → Esta es obtenida en base a la tabla 4.3.

Otras opciones encontradas en el encabezado IPv6, tales como *Hop-by-Hop*, *Routing*, etc., deberán ser ignoradas, puesto que el intentar realizar la conversión del valor de alguna de estas opciones, puede causar inconsistencia en el encabezado IPv4.

En el caso de que se encuentre un encabezado *Routing* presente, este paquete deberá ser desechado y un mensaje ICMPv6 “problema de parámetros, encontrado campo del encabezado erróneo” deberá ser enviado a la dirección IPv6 fuente.

El método de *tunneling* descrito en este documento, no encapsula opciones de encabezado de ruteo IPv6 ni extensiones de encabezado tales como la trayectoria de ruta predefinida, aún más, opciones del encabezado IPv4 no son tomadas en consideración.

Aún nos falta calcular el campo *Checksum*, para anexar este valor en el encabezado del paquete IPv4 a construir. Esto se realiza mediante el pseudo código mostrado en la figura 4.3, que utiliza el método de *complemento a uno* como algoritmo de cálculo.

Otro punto importante, es analizar cuales campos en el encabezado IPv6 deberán ser modificados si es que se requiere.

⁴ En el caso de que el encapsulador sea un ruteador, este deberá decrementar este campo en IPv6 antes del encapsulamiento o el campo *TTL* después del encapsulamiento.

Se sabe que el máximo número de saltos [NaT+98] es de 255 en IPv6 y el valor por defecto es de 64 saltos, así que normalmente este último es el valor que tendrá el campo *TTL*, pero si el paquete atraviesa uno o más ruteadores, este campo se deberá decrementar por uno en cada salto.

Tanto en el encabezado IPv6 como en el encabezado IPv4 del paquete encapsulado, este valor deberá modificarse simultáneamente para evitar conflictos con distintos valores del campo *Hop Limit* en IPv6 y el campo *TTL* en IPv4.

Campos adicionales no es necesario modificarlos en la parte del encabezado IPv6, puesto que no se manejan aspectos tales como *trayectoria de ruta predeterminada* entre otros.

```
Packet structure {  
    Heading structure;  
    message[size of the message];  
}packet;  
checksum(length, *buffer) {  
    add all the 16 bits words;  
    if a carrying bit exists  
        add to the sum;  
    add to the carrying;  
    add again to the carrying;  
    evaluate the complement to one;  
    return to a value of 16 bits  
}
```

Figura 4.3. Pseudo código para calcular el valor del campo *checksum*.

El protocolo IPv6 ha sido diseñado para que el campo *checksum* de los encabezados *TCP* y *UDP* no sea afectado por los distintos tipos de *tunneling* realizados [NoE00], en particular la técnica que aquí se presenta en este documento, también entra dentro del concepto de las técnicas de *tunneling* existentes.

De esta manera, el encapsulador no necesita modificar los encabezados *TCP* y *UDP*. Las únicas excepciones son paquetes IPv4 *UDP no fragmentados* los cuales necesitan tener un *checksum* calculado debido a que este último es requerido por *UDP* en IPv6.

También ICMPv6 incluye un *checksum*, pero este no está presente en ICMPv4, por lo tanto, este campo en los mensajes ICMP solo necesita ser modificado por el encapsulador. Además los mensajes de error ICMP incluyen el encabezado IP como parte de la carga, por lo tanto, el encapsulador necesita sobrescribir esta parte del paquete para asegurarse que el receptor sea capaz de entender en encabezado incluido [NoE00].

Una vez definidos los valores de los campos en IPv4, ya se tienen las herramientas necesarias para la construcción del encabezado y del paquete IPv4. La información o datos, es el paquete IPv6 que será enviado.

Se procede a crear los socket's para construir el paquete, tomando en consideración que la aplicación que realiza el encapsulamiento debe seleccionar las direcciones fuente y destino tanto del paquete como del nodo encapsulante y desencapsulante. El proceso de anexar el encabezado se muestra en el pseudo código de la figura 4.4.

En este punto ya se tiene el encabezado IPv4 con sus respectivos valores asignados a los campos y se ha enviado el paquete, esto es se ha realizado el proceso de encapsulamiento.

```
main{
  structure definitions {
    Heading structure;
  }
  definitions;
  create the socket raw
  link the socket
  connect
  read information (IPv6 packet)
  header IPv4 → field of header IPv6
  define heading options
  add IPv4 heading
  send information to destination
  close socket
}
```

Figura 4.4 Pseudo código para construir el encabezado IPv4.

Además de anexar el encabezado IPv4, el nodo encapsulante debe manejar otros elementos, entre los cuales se encuentran:

- Determinar cuando fragmentar.
- Como manejar los errores ICMPv4 (ICMP para el protocolo IPv4) de los ruteadores y los errores ICMPv6.
- Modificar en caso de ser necesario el encabezado IPv6 cuando el paquete tiene que viajar a través de dos o más ruteadores.

La fragmentación es otro elemento importante a considerar durante el encapsulamiento. Es necesario recordar que una de las diferencias importantes entre IPv4 e IPv6 es el proceso de descubrimiento del MTU en cual es obligatorio en IPv6 pero es opcional en IPv4. Esto implica que los ruteadores IPv6 nunca fragmentaran un paquete, solamente el emisor puede realizar esta acción.

Lo anterior nos lleva a que el emisor deberá asegurarse que el paquete no exceda el MTU en la red IPv6, de lo contrario el proceso de fragmentación deberá realizarse.

Si el tamaño del paquete es menor de los 1280 bytes, este nunca será fragmentado, ya que IPv6 lo garantiza con un paquete de estas características, a diferencia de IPv4 el cual lo garantiza con un paquete de tamaño de 68 bytes. En caso de que el paquete supere este tamaño, el emisor deberá incluir un encabezado *fragment* IPv6 para indicar que este permite la fragmentación.

Las reglas anteriores aseguran que los paquetes deberán ser fragmentados ya sea por el emisor o por el nodo encapsulador en caso de encontrarse este con un encabezado *fragment* opcional en el paquete IPv6. Si este último realiza el proceso de fragmentación, entonces deberá llevar los 16 bits de más bajo orden sin modificación alguna hasta el receptor para asegurar el correcto reensamble.

Existen algunas diferencias entre IPv4 e IPv6 en el área de fragmentación, y el mínimo enlace MTU que afecta el encapsulamiento. Un enlace IPv4 tiene un MTU de 68 bytes y un enlace IPv6 tiene un MTU de cuando menos 1280 bytes.

Por lo tanto esto hace imposible descubrir la MTU en una trayectoria punto a punto que incluya un encapsulador de por medio, debido a que un nodo IPv6 puede recibir paquetes con un mensaje ICMPv6 de *packet too big* originado por un ruteador IPv4 que reporta un MTU menor de 1280 bytes.

Sin embargo en IPv6 se requiere que el nodo maneje el mensaje ICMPv6 obtenido reduciendo la MTU al mínimo de 1280 bytes y además incluyendo un encabezado *fragment* con cada paquete.

Cuando la MTU cae por debajo de los 1280 bytes, el nodo emisor IPv6 debe enviar paquetes de 1280 bytes incluyendo en encabezado *fragment* que serán fragmentados por los ruteadores IPv4 a lo largo de la trayectoria por la cual están siendo trasladados.

Existen otros elementos a manejar en cuanto a fragmentación se refiere, tales como utilizar PMTU (Path MTU) para hacer la fragmentación UDP óptima, estos se mencionan en el apéndice B.

En los párrafos anteriores, se definió detalladamente el proceso de encapsulamiento con sus respectivos valores, ahora el siguiente paso es realizar el proceso inverso.

Para lograr este propósito, es necesario tener en consideración que el paquete IPv4 al cual se le quiere eliminar su encabezado, lleva como carga el paquete IPv6 “original”, aún más, el nodo desencapsulador es el responsable de esta tarea, no el nodo destino, salvo que sean el mismo.

El nodo desencapsulador puede ser un ruteador, aún más, este debe tener otras funciones, como se vera más adelante, por ejemplo, en caso de que este no sea el nodo destino, el mismo deberá encargarse de retransmitir el paquete al nodo final, pero ya en una red IPv6. La información correspondiente para retransmitir el paquete IPv6 viene incluida dentro del mismo, recuerdese de la tabla 4.3, que el paquete encapsulado tiene dos direcciones de envío, una es la del nodo desencapsulador y otra es la del nodo destino.

Para realizar esta tarea, se tomo en cuenta que ya no se van a modificar los campos del encabezado, además, la idea es realizar este proceso de tal manera que se obtenga un paquete IPv6 idéntico al el enviado por el nodo emisor, salvo por el campo *Hop Limit*, el cual es probable que haya sido modificado.

Teniendo esta precauciones, se presenta en la figura 4.5 el pseudo código para eliminar el encabezado del paquete IPv4 recibido.

Este proceso de eliminar el encabezado, o mejor aún, el proceso el general de *tunneling* es unidireccional, así, para lograr una comunicación en ambos sentidos, debe realizarse esta técnica de manera bidireccional, de tal forma que el punto de entrada del túnel en un nodo es el punto de salida del túnel en el otro nodo.

Este mecanismo de túnel bidireccional es logrado construyendo dos mecanismos de túnel unidireccionales en cada punto del túnel.

4.5 Resultados obtenidos.

Se realizaron varias pruebas para lograr la comunicación entre los nodos finales, inicialmente se lograron enlaces *orientados a conexión* (sock_stream) y *sin conexión* (sock_dgram) entre nodos IPv4 y nodos IPv6 exclusivamente.

```

main{
  structure definitions {
    Heading structure;
  }
  definitions;
  create the socket raw
  link the socket
  listen
  accept
  read information (encapsulated packet)
  process application
    if the destination is different from the hostname
      send the packet from destination
    else
      eliminate heading
  close socket from the emission side
}

```

Figura 4.5. Pseudo código para eliminar el encabezado IPv4.

Posteriormente, se realizaron las primeras pruebas, encontrándose algunas dificultades en cuanto a comunicación se refiere entre los nodos usando la técnica de *tunneling*, debido al hecho de que al modificar el encabezado del paquete, se debe tener acceso a la capa IP, sin embargo, con los sockets tipo *TCP* y *UDP* no se tiene acceso al interior de los paquete para modificarlos.

Lo anterior, nos llevo a utilizar el *sock_raw*, el cual ofrece más flexibilidad. Para lograr esto, se debe tener acceso a la capa IP, sin embargo, con los sockets del tipo *TCP* y *UDP* no fue posible, como se menciono anteriormente. El *socket raw* ofrece mucha flexibilidad, se abren los protocolos de gestión de mensajes (de error). Los protocolos de alto nivel cierran la posibilidad del envío de paquetes ICMP. Recuerdese que los *sockets raw* son un nivel abstracto de la capa de red.

Este tipo de sockets tiene algunas limitaciones, la pérdida de confiabilidad por ejemplo (de manera similar a *UDP*), con lo cual el núcleo pasa un *paquete raw* a todos los *sockets raw* iguales, y por lo tanto, no es posible determinar si más de un *socket raw* tiene el mismo número de protocolo y también tiene algunas ventajas, puesto que es posible modificar “todos” los campos del encabezado en la capa IP (capa 3 del modelo OSI).

Al proceder a agregarle el encabezado a un paquete, se encontraron dificultades en la realización de los programas, por ejemplo, los enlaces no reunían las características necesarias en cuanto a los campos agregados, es decir, se creaban campos con valores que no correspondían a lo esperado.

Superando estos detalles, finalmente fue posible realizar enlaces con las características necesarias. En estos enlaces solo se enviaron mensajes de texto. De tal forma de no realizar fragmentación alguna. Debido a que ello implica, primero descubrir el MTU, segundo fragmentar el paquete y finalmente, enviar cada paquete fragmentado a su destino, con la información suficiente para el reensamble.

El hecho de descubrir el MTU, es difícil, en el sentido de que debido a que se manejan dos redes con distintas características (IPv4 e IPv6), complica en gran medida el cálculo de este parámetro. Para lograrlo, éstas deben satisfacer requisitos especiales.

Todas las pruebas efectuadas en la red que manejan paquetes encapsulados, fueron enviadas sobre esta misma sin utilizar el protocolo de descubrimiento del MTU.

La forma de comprobar que realmente estos enlaces cumplían las características necesarias, fue, realizando un programa que *espiara* los paquetes transmitidos en la red, y mostrara la información en pantalla de todo el paquete en formato hexadecimal. Esta herramienta fue bastante útil, en el sentido de que realmente se comprobó la veracidad de lo obtenido.

En la figura 4.6 se muestra la salida obtenida con este programa “*espía de paquetes*” que se utilizó para las pruebas. Cabe hacer notar que este “*espía*” fue una herramienta indispensable, debido a que en un principio no se contaba con alguna herramienta que se adecuara a nuestras necesidades, lo cual nos llevó a realizar el diseño de este programa. Es cierto que existen herramientas que realizan tareas similares, éstas están fuera del alcance, por ejemplo, un analizador de protocolos.

En esta figura, se observa que el paquete contiene varios campos, los cuales se describen a continuación, tomando en consideración la numeración observada en esta:

- El campo 1, es la trama de un paquete que cruza una red IPv4.
- El campo 2, es el encabezado IPv4 del mismo paquete, teniendo como datos un paquete con distinto protocolo.
- El campo 3, es el encabezado del paquete IPv6. En este campo, es posible modificar el encabezado *Hop Limit* cuando este paquete atraviesa uno o más ruteadores.
- El campo 4, con los datos del paquete IPv6, en este caso es un mensaje, teniendo precaución de no sobrepasar el MTU, como se mencionó anteriormente.
- El campo 5, insertado al final de los datos de manera predeterminada.

Finalmente, se pudo comprobar el satisfactorio encapsulamiento realizado. Esta técnica, presenta algunas ventajas y desventajas comparada con otras técnicas similares. En las siguientes secciones se menciona con detalle este punto.

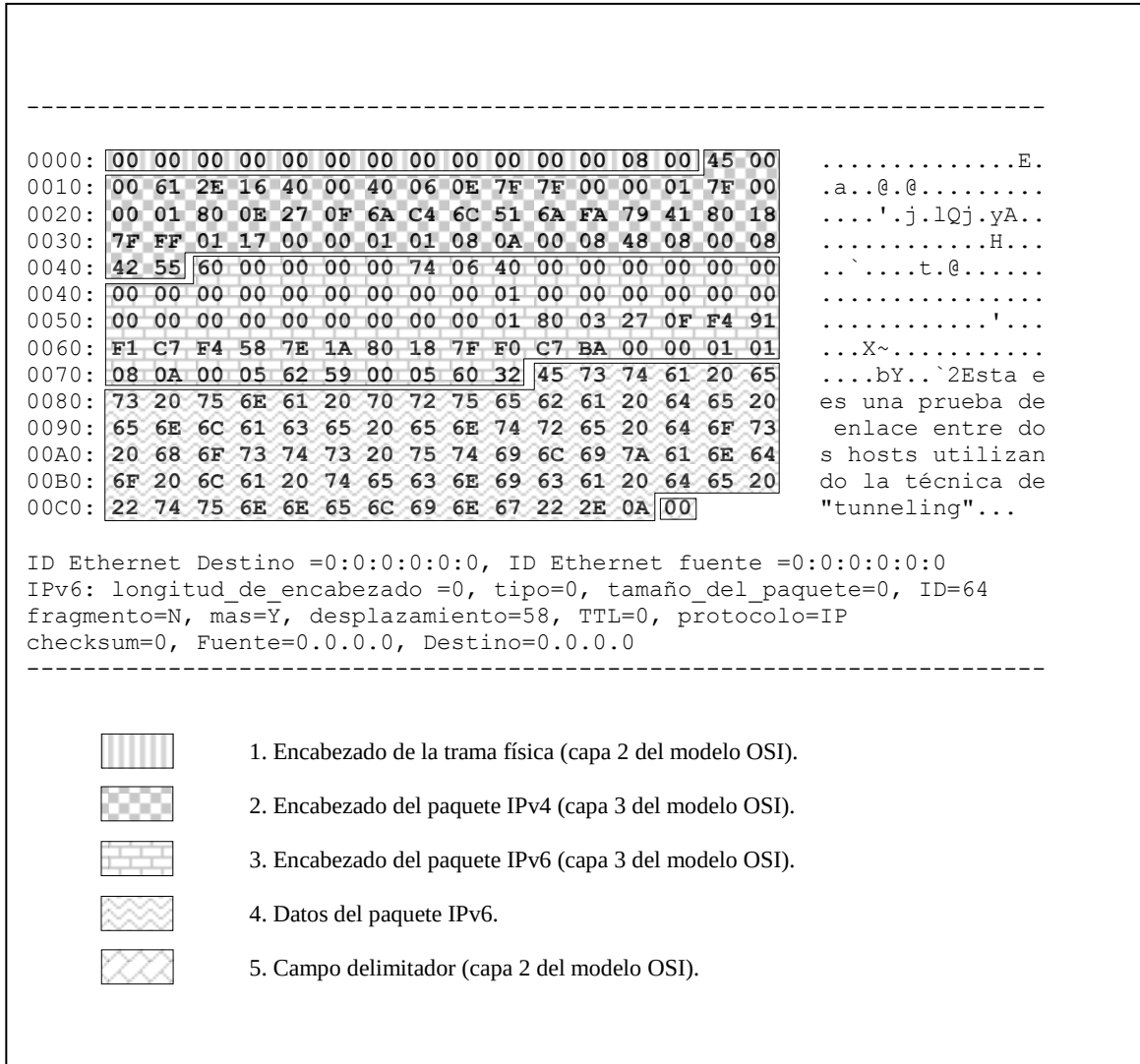


Figura 4.6. Ilustración de un paquete encapsulado.

4.6 Aspectos relevantes.

Existen algunos puntos importantes al respecto de la técnica presentada en este documento, sen iniciará esta sección presentando las ventajas y desventajas de la técnica presentada aquí:

- Ventajas:
 - Tipo de direcciones. Mediante esta técnica, es posible manejar distintos tipos de direcciones, tales como direcciones unicast, *anycast* y *multicast*, esta técnica no esta restringida solo a direcciones del tipo IPv4 con compatibilidad IPv6 por ejemplo como en otras técnicas.
 - Configuración. El usuario no tiene que realizar configuración alguna.
 - Sencillez y simplicidad en el uso de esta técnica.
 - Compatibilidad con el soporte para DNS. Esto es debido a que el túnel utiliza direcciones IPv4 para transmitir el paquete encapsulado dentro de la red IPv4.
 - El soporte para Dual Stack no es indispensable.
 - Otros métodos de encapsulamiento requieren configuración adicional en los host y ruteadores para lograr su objetivo y el método aquí presentado no requiere de tal configuración en los host.
- Desventajas:
 - Pérdida de fiabilidad debida al tipo de sockets utilizado. Esta característica viene implícita en los sockets raw.
 - Las direcciones de entrada y salida del túnel dependen de la configuración instaurada en este, esto es, si en algún momento es errónea la información proporcionada al encapsulador, este no podrá enviar el paquete a su destino.
 - Falta de mecanismos de seguridad implementados en esta técnica, tales como encriptación.

Hasta este momento ya se ha analizado completamente la técnica aquí presentada, ahora se procederá a analizar que requisitos adicionales pueden implementarse con objeto de mejorarla.

Algunas propuestas que se pueden realizar para mejorar esta técnica, con objeto de hacerla una herramienta útil en el marco de la investigación o aún más en el marco empresarial, se presentan en los siguientes puntos:

- El aspecto de seguridad es un elemento fundamental, el cual es necesario implementar con objeto de hacer a esta técnica confiable. Estos mecanismos de seguridad, pueden diseñarse sobre el marco base de esta herramienta, por ejemplo, agregando encriptación a los programas diseñados.
- Otro elemento indispensable a implementar, es el de *Calidad de Servicio*, para manejar información en tiempo real. Aprovechar al máximo el ancho de banda para darle el rendimiento óptimo a esta herramienta.
- El manejo de direcciones de tipo *multicast* debe implementarse. Para lograr esto, el ruteador debe utilizar el protocolo MLD para descubrir los nodos a los cuales es necesario enviar este tipo de paquetes. Así, para que un nodo envíe un paquete *multicast*, este deberá dirigirlo al ruteador, el cual tendrá que verificar el tipo de dirección con objeto de darle un tratamiento especial a estos paquetes.

En el caso de los paquetes con destino *multicast*, el ruteador deberá encapsular los paquetes y posteriormente, desencapsular estos antes de sean transmitidos a los nodos con direcciones *multicast* (para ello, es mejor contar con dos ruteadores configurados para realizar el encapsulamiento). No se debe olvidar que el ruteador tiene una lista de los nodos con dirección *multicast*.

4.7 QoS y Consideraciones de seguridad.

Es posible implementar QoS a un flujo de datos particular utilizando los mecanismos de *traffic control*, los cuales incluyen un *clasificador de paquetes*, *control de admisión*, y un *planeador de paquetes* o algún otro mecanismo dependiente de la capa 2 del modelo OSI. El *clasificador de paquetes*, determina la clase de QoS y aún más, la ruta para cada paquete y el *planeador de paquetes* es el responsable de mantener la QoS.

Así, para reservar recursos a un paquete con solicitud de QoS, es necesario crear dos módulos: de *control de admisión* y de *políticas de control*. El primero determina si el nodo tiene suficientes recursos para proporcionar la QoS solicitada, y el segundo determina si el usuario tiene permisos administrativos para realizar la reservación de estos recursos. En caso de ser afirmativa la creación de estos dos módulos, los parámetros son agregados al paquete para que este obtenga la QoS deseada, ó en caso contrario, se regresa un mensaje de notificación de error a la aplicaron solicitante.

El uso de la herramienta aquí desarrollada, no introduce nuevos elementos de seguridad más allá de la seguridad que esta presente es los protocolos IPv4 e IPv6 y en los protocolos de ruteo, los cuales son utilizados para hacer que el paquete alcance su destino. Aún más, la seguridad proporcionada con el método de tunneling, no va más allá de la seguridad ofrecida por el protocolo *UDP*, puesto que esta es similar a la seguridad ofrecida por los *sockets raw*. Este es un servicio del mejor esfuerzo y esta implementado como servicio por defecto en la red.

Un túnel IPv6 puede mejorar su seguridad, mejorando la seguridad entre la trayectoria del punto de entrada y salida del mismo. En este camino, es posible aplicar algoritmos de seguridad en el paquete y agregarlos como parte del encabezado del paquete.

Estos nuevos elementos de seguridad pueden ser ofrecidos agregando código extra a la aplicación, y para lograrlo, es necesario profundizar en los mecanismos intrínsecos, tales como la *autenticación del encabezado*, la cual es especificada para incluir el campo *Identification* en IPv4 y la función de encapsulamiento deberá encargarse de calcular este campo en paquetes recibidos que están siendo encapsulados, debido a que no es posible para un nodo final IPv6 calcular este campo en paquete recibidos que han sido desencapsulados de paquetes IPv4.

Otro elemento útil en el ámbito de seguridad que se puede implementar, es utilizar *ESP* (Encapsulating Security Payload) *en modo túnel*, el cual es diseñado para proveer una mezcla de servicios de seguridad en IPv4 e IPv6, y consta de agregar un encabezado en la capa 3 del modelo OSI (capa en la cual trabaja el túnel aquí presentado) con sus respectivos parámetros de seguridad intrínsecos.

El hecho de mencionar algunos elementos de QoS y seguridad en esta parte del documento, es con el fin de proporcionar al lector una idea de las posibles soluciones que existen en materia, y con la cual es completamente factible mejorar la herramienta aquí presentada.

Conclusiones.

El proyecto que aquí se desarrolló, realiza una caracterización del comportamiento de la técnica de *tunneling* enviando distintas clases de paquetes que tienen que atravesar redes híbridas, también presenta al usuario una aplicación que habilita la comunicación entre redes IPv6 atravesando redes IPv4; además, proporciona una aplicación para lograr este enlace, evitando realizar configuración adicional en el nodo cliente de la subred IPv6. Basta con proporcionar la dirección IPv6 del nodo receptor como argumento y el mensaje para que éste sea enviado a través de la subred.

Para realizar este proyecto, es indispensable tener un conocimiento amplio sobre los protocolos IPv4 e IPv6 y los métodos de *tunneling* existentes, así también del manejo de los sockets utilizando el lenguaje de programación C.

Las pruebas iniciales se realizaron sin mayor dificultad. Se lograron enlaces utilizando únicamente el protocolo IPv4 e IPv6 y luego se realizaron otras pruebas utilizando ambos protocolos simultáneamente. En estas pruebas se comprobó que efectivamente se cumplen los requisitos de conectividad en la subred que se diseñó, tal como era de esperarse. Además, se comprobó que efectivamente la configuración realizada en el ruteador satisface las características deseadas.

Debido a la complejidad existente en los apuntadores a estructuras, banderas de codificación y de reconocimiento, fue necesario investigar ampliamente al respecto, así también sobre el marco conceptual mediante el cual se manejan las bibliotecas de sistema en el Sistema Operativo *Linux SuSE*.

Al iniciar la implementación, se tuvieron algunos problemas en las conexiones con los socket's para agregar el encabezado, puesto que realmente no se obtenía el comportamiento deseado.

Los primeros programas no agregaban el encabezado deseado, inclusive no era posible modificar los encabezados de paquetes enviados sin utilizar el método de *tunneling*. Así que fue necesario iniciar los programas intentando modificar los encabezados de paquetes IPv4 en la red IPv4 únicamente.

Después de varios intentos, se lograron tales resultados. Posteriormente, se procedió a utilizar el protocolo IPv6 para nuestras pruebas, inicialmente modificando el encabezado de paquetes IPv6 en redes IPv6 únicamente, logrando con éxito el objetivo.

Una vez logrando este primer objetivo, se realizaron las primeras pruebas encapsulando un paquete. Para ello se diseñaron varios programas con el fin de agregar el encabezado, primero se agregaron los campos básicos del protocolo IPv4, según se describe en el capítulo 4.4, posteriormente, se procedió a crear los socket's para construir el paquete.

Fue necesario seleccionar adecuadamente los puntos de entrada y salida del túnel para que de manera correcta se realizara el encapsulamiento, lo cual lo efectúa el programa que hace el tunneling.

En este momento del desarrollo del proyecto, ya se contaba con una herramienta que realizaba el encapsulamiento, ahora lo importante era verificar el proceso que se realizaba correspondía a lo que efectivamente se esperaba. Para ello se tuvo que diseñar otra invaluable herramienta que “escuchara los paquetes” y que presentara éstos tal como transitaban en la red. Esta última herramienta fue un elemento importante en el proceso de depuración de los programas.

Se decidió crear esta herramienta “escuchador de paquetes”, con el fin de que realizara los objetivos deseados, puesto que al utilizar alguna herramienta de propósito general, habría que modificarla ó adaptarla a nuestras necesidades. Fue así como pudimos comprobar que efectivamente el método desarrollado realizaba el encapsulamiento de manera correcta y como era de esperarse.

La herramienta arriba mencionada, cuenta con algunas características importantes, entre las cuales se encuentran:

- Filtro para paquetes con un protocolo particular, IPv4 e IPv6 y ambos.
- Filtro para paquetes de un tipo particular, tales como es ICMP, UDP, TCP e IP.
- La interface de red sobre la cual trabaja, no necesita estar en modo *promiscuo*.
- Debido a la utilización de *socket_raw*, es posible tener acceso a cualquiera de las capas inferiores del modelo OSI.
- Herramienta flexible con capacidad de adaptarla a nuestras necesidades particulares.

Finalmente, fue posible comprobar la veracidad del método de encapsulamiento con la herramienta antes mencionada y que efectivamente se satisfacían los requisitos indispensables descritos para la técnica que aquí se desarrolló.

Una vez realizado el *tunneling*, se encontraron algunos puntos importantes a considerar respecto a la técnica aquí desarrollada, además de algunos elementos importantes para mejorar la técnica aquí propuesta, entre los cuales es necesario considerar los siguientes:

- Aspectos importantes del método de *tunneling* aquí desarrollado:
 - El proyecto aquí realizado nos presenta un análisis del comportamiento de la técnica de *tunneling* manejando distintas clases de paquetes, los cuales deben cruzar redes con distintas características e interactuando con los protocolos IPv4 e IPv6.
 - La comunicación entre nodos cuenta con los mecanismos de seguridad presentes en los protocolos IPv4 e IPv6 y de los sockets raw.

- El envío de paquetes encapsulados a través de la red IPv4 cuenta con características por defecto como cualquier paquete IPv4, pero éstas fácilmente se pueden mejorar agregando opciones extra a los campos correspondientes con el fin de proporcionar servicios de QoS para un tráfico particular, puesto que ya se tienen las bases teóricas, prácticas y herramientas para lograr este objetivo.
 - El resultado de la implementación de los servicios de QoS es obtenido en la red IPv4 únicamente. Si es necesario manejar necesidades de tráfico particular en la red IPv6, éstas deberán ser agregadas antes del encapsulamiento. De este punto surge una propuesta interesante, la cual es que dada una necesidad de tráfico particular inicialmente en la red IPv6, esta se pueda extender a la red IPv4, es decir, *trasladar* las características de QoS (por ejemplo) al paquete encapsulado y mantener éstas a través de la red IPv4.
- Propuestas para mejorar la técnica aquí descrita:
 - Verificar el envío de errores. Sobre este punto, es necesario considerar que el manejo de errores debe realizarse en la red IPv6, esto es, el nodo emisor debe recibir la información de los errores detectados durante el envío del paquete. Ésto es posible realizarlo haciendo un programa con *sock_dgram* (en la capa 4 del modelo OSI).
 - Verificar que la transmisión de paquetes se realice utilizando un destino *multicast*. Para realizar el encapsulamiento con este tipo de direcciones, es necesario modificar la forma en que se asigna el nodo destino en un paquete. Consideramos que investigar mas a fondo al respecto en este punto nos proporcionaría una invaluable herramienta puesto que se eliminan las restricciones impuestas por el uso de direcciones unicast. Aún más, el diseño utilizando direcciones *multicast* puede ser fácilmente escalado a direcciones *anycast* o viceversa. Al lograr este propósito, se estaría rompiendo la barrera impuesta por la mayoría de las técnicas comunes que utilizan solamente direcciones unicast.
 - Consideraciones de seguridad deberán ser agregadas para proveer una herramienta que supere los requerimientos de confiabilidad necesarios. Tal es el caso de utilizar el campo *traffic control* para clasificar los paquetes que necesiten un manejo especial. En la sección 4.7 se mencionan algunos puntos importantes a considerar al respecto de QoS.
 - La fragmentación es otro elemento importante para implementar en el método de tunneling aquí descrito. Debido a la diferencia del MTU en redes IPv4 e IPv6, es necesario investigar ampliamente al respecto. Para resolver lo anterior, se propone que sea el nodo encapsulador quien maneje la fragmentación, considerando que el nodo emisor también realiza una fragmentación si el paquete supera el MTU de una red IPv6. Esto es, el nodo encapsulador deberá fragmentar de acuerdo a las necesidades de la red IPv4 los paquetes, independientemente de si el nodo emisor haya fragmentado un paquete, y el nodo desencapsulador deberá realizar el proceso de reensamble. Esto proveerá algo de sobrecarga a los nodos encapsuladores, pero se

considera que vale la pena dejar esta labor a los mismos, y al mismo tiempo solucionar el problema de la diferencia de MTU's en las diferentes redes.

Se considera que al profundizar en el proceso de fragmentación se encontraran interesantes resultados, aún más, es posible proponer una mejora al respecto sobre los métodos de fragmentación con *tunneling* existentes.

Al término de la realización de este proyecto, satisfactoriamente se puede concluir algunos elementos importantes obtenidos, entre los cuales se encuentran:

- Lograr la presentación de una **Publicación Internacional** del proyecto realizado, titulada “**IPv6 ENCAPSULATING TECHNIQUES**” patrocinada por IASTED CST 2003 International Conference Computer Science and Technology, en Mayo de 2003 [LoM+03]. Lo cual nos lleva a contribuir con el prestigio de la Universidad a la cual orgullosamente pertenecemos.
- Contar con elementos suficientes para poder implementar o mejorar técnicas de comunicación en general.
- Contar con una base sólida de conocimiento en los protocolos IPv4 e IPv6 y en general en redes.
- Lograr una meta de superación personal ampliamente anhelada entre otras, para ser competitivos en el mercado laboral.

Finalmente es seguro de que las innumerables experiencias obtenidas durante la realización del proyecto contribuyeron ampliamente en mejorar la formación Universitaria obtenida durante el transcurso de los estudios. Aún más, queda una importante tarea por realizar, la cual es continuar actualizándose en materia, puesto que el desarrollo e investigación continúan a pasos enormes, y un elemento de competitividad laboral es permanecer a la vanguardia.

Apéndice A. Estructura de direcciones IPv6.

En este apéndice, se profundizará un poco en los tipos de direcciones existentes y las extensiones de encabezado más comunes.

A.1 Direcciones IPv6.

Esta estructura maneja múltiples tipos de direcciones, entre las cuales se encuentran las siguientes:

La dirección no especificada.

La dirección “0:0:0:0:0:0” o bien “::” es llamada dirección no especificada. Esta nunca puede ser asignada a un nodo, esta indica la ausencia de una dirección.

La dirección no especificada no debe ser utilizada como dirección de destino en paquetes IPv6 o en encabezados de ruteo IPv6. Esta dirección es utilizada como dirección fuente por un host mientras realiza *configuración automática*.

La dirección “loopback”

La dirección *unicast* “0:0:0:0:0:1” o bien “::1” es llamada la dirección “loopback”. Esta puede ser utilizada por un nodo para enviar paquetes a si mismo. Esta no puede ser asignada a una interfaz física. Esta no debe ser utilizada como dirección de destino. Puede asociarse como una interfaz virtual (es decir, la interfaz loopback).

Un paquete con dirección destino ::1 no debe salir de esa interfaz y los ruteadores nunca deberán transmitir estos paquetes.

Las direcciones NSAP.

La dirección NSAP es la información que el proveedor de servicio de red OSI necesita para identificar un punto de acceso de servicio de red particular.

Es posible el mapeo de direcciones NSAP en direcciones IPv6. Se sugiere que aquellos interesados en utilizar el esquema de direccionamiento NSAP tengan que rediseñar su plan de direccionamiento nativo IPv6 [MoJ+90]. Existen otros cuatro mecanismos para el soporte de direccionamiento OSI NSAP en una red IPv6 [BoJ+96], estos son:

- Mapeo de direcciones NSAP restringidas en direcciones IPv6 de 16 bytes.
- Direcciones NSAP truncadas para ruteo, dirección NSAP completa en la opción IPv6.
- Direcciones IPv6 trasladadas como direcciones OSI.

El direccionamiento IPX.

Las direcciones IPX (Internetwork Packet Exchange) deberán ser mapeadas en direcciones IPv6 con un formato que inicia con un prefijo de formato de 7 bits. En la figura A.1 se muestra este mapeo.

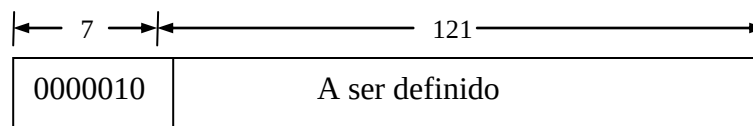


Figura A.1 Formato de una dirección IPX.

Direcciones unicast globales agregables.

Este formato de direcciones está diseñado para soportar el alcance basado en proveedor y el nuevo tipo de alcance llamado de intercambio. Esta combinación permite un alcance de ruteo eficiente para los sitios que se conectan mediante el proveedor y los sitios que utilizan el intercambio. Los sitios deberán seleccionar mediante que formato de dirección se conectan.

Para IPv6 la jerarquía de direcciones agregable, está organizada en tres niveles como se documenta en [HiR+98b]:

1. Topología pública. Es una colección de proveedores y de intercambio que proveen un servicio de *tránsito* de Internet público a nodos fuera del sitio.
2. Topología de sitio. Es local a un sitio específico u organización, pero no provee servicio de *tránsito público* a nodos fuera de este sitio.
3. Identificador de interfaz. Provee una identificación única para interfaces en un enlace específico.

Direcciones unicast de uso local.

Existen dos tipos definidos de direcciones *unicast* de uso local:

- *Enlace local*. Direcciones de esta clase pueden ser utilizadas en la red física en la que la interfaz del host está conectada. Las direcciones de *enlace local* son utilizadas para direccionamiento en un solo enlace, para propósitos tales como *configuración automática* de dirección, descubrimiento de vecinos o cuando no hay ruteadores presentes.
- *Sitio local*. Direcciones de esta clase no pueden ser ruteadas a Internet. Son para uso de un solo sitio. Están diseñadas para ser utilizadas para direccionamiento dentro de un sitio sin la necesidad de un prefijo global. Estas redes son equivalentes a las redes de uso privado IPv4.

Los ruteadores no deben transmitir cualquier paquete con dirección fuente o destino de *enlace local* a otros enlaces. Así mismo los ruteadores no deben transmitir cualquier paquete con dirección fuente o destino de *sitio local* a otros sitios.

Direcciones de host IPv6 con capacidad IPv4.

Existen dos tipos de direcciones de host con capacidad de proporcionar un enlace directo hacia una red IPv4, las cuales son:

- Direcciones *IPv6 con compatibilidad IPv4*. Son utilizadas cuando tráfico IPv6 se necesita encapsular y enviar sobre una red IPv4.
- Direcciones *IPv6 mapeadas a IPv4*. Son utilizadas cuando un host IPv6 necesita comunicarse con un host IPv4.

A2. Las extensiones del encabezado.

En el caso de que no exista encabezado, esto se indica con un valor de “59” en el campo *Next Header* del paquete IPv6 o de cualquier extensión de encabezado. Ello indica que ya no hay más encabezados.

Es común que solo exista el encabezado IP sin extensiones de encabezado. Este es el caso por defecto.

El diseño de IPv6 simplifica el encabezado existente de IPv4 colocando muchos de los campos existentes y poco usuales de IPv4 en campos opcionales de encabezado. Con esta medida, se logra evitar sobrecarga en el manejo de paquetes del procesador. Es necesario notar que las extensiones de encabezado, deberán ser múltiplos de 8 bytes en longitud, para mantener el alineamiento de *ocho bytes* en los encabezados subsecuentes.

Un paquete IPv6 puede tener cero o más extensiones de encabezado ver figura A.2, cada uno identificado por el campo *Next Header* del encabezado anterior.

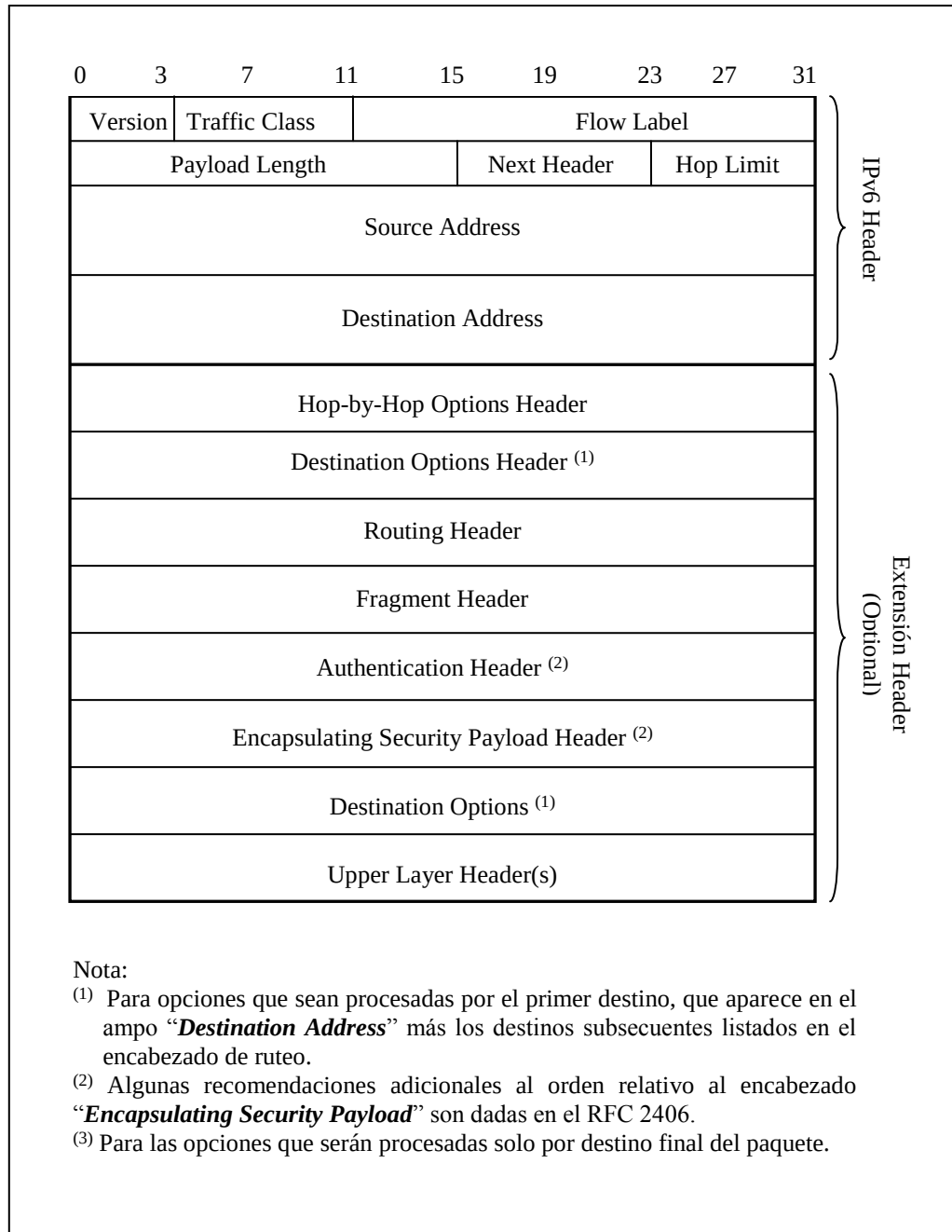


Figura A.2. Formato de un paquete IPv6 con encabezados opcionales.

La longitud de cada encabezado varia dependiendo del tipo, pero siempre es un múltiplo de 8 bytes. Existe un número limitado de extensiones de encabezado, y solo deberá aparecer una vez en el paquete, a excepción del encabezado *Destination Options*, el cual puede aparecer más de una vez. Además estos encabezados deben seguir un orden

específico aunque los nodos IPv6 que reciban el paquete no es necesario que verifiquen este orden.

El orden es importante para hacer más eficiente el procesamiento en ruteadores intermedios, los cuales generalmente se interesan en el encabezado *Routing Header* y el encabezado *hop-by-hop*.

Los ruteadores IPv6 no agregan extensiones de encabezado a un paquete, pero en vez de esto, ellos podrán encapsular el paquete recibido dentro de un paquete IPv6 que ellos mismos crean.

Las extensiones de encabezado deben ser procesadas en el orden estricto en que ellas aparecen en el paquete. Cuando más de una extensión de encabezado sea agregada al paquete, se recomienda que éstas sigan un orden predefinido.

Orden que deben seguir las extensiones del encabezado: abc

- Encabezado IPv6.
- Opciones de encabezado *Hop-by-Hop*.
- Opciones de encabezado *Destination* ⁵.
- Encabezado *Routing*.
- Encabezado *Fragment*.
- Encabezado *Authenticación*.
- Encabezado *Encapsulating Security Payload* ⁶.
- Opciones de encabezado *Destination* ⁷.
- Encabezado del protocolo de la capa superior (ejemplo TCP, UDP, etc.).

Dos de las extensiones de encabezado, la opción *Hop-by-Hop* y la opción *Destination* pueden llevar una o más opciones que aún más identifiquen los parámetros en la operación de red.

Las opciones de encabezado *Hop-by-Hop*.

Esta lleva información opcional que debe ser examinada por cada nodo junto con la trayectoria del paquete enviado. Este encabezado contiene dos campos más opciones, ver figura A.3.

El primer campo es de 8 bits e identifica el encabezado inmediatamente siguiente a la opción de encabezado *Hop-by-Hop*.

⁵ Solo para opciones que serán procesadas por el primer destino que aparece en el campo *Destination Address* más los subsecuentes destinos que aparecen listados en el encabezado *Routing*.

⁶ Existen recomendaciones adicionales a estos encabezados [KeS+98a].

⁷ Se aplica a opciones que solo serán procesadas por el destino final del paquete.

El segundo campo es de 8 bits e identifica la longitud de la extensión de encabezado sin contar los primeros 8 bytes. El campo *Options* es de longitud variable, de tal manera que complete este encabezado para que sea un múltiplo de 8 bytes en longitud.

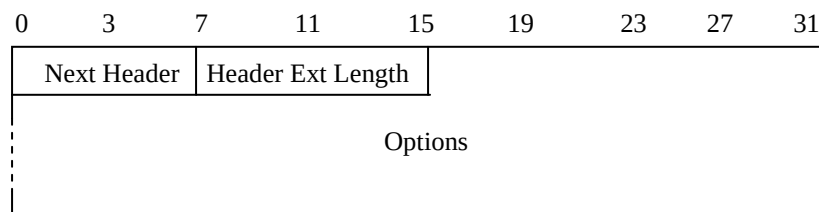


Figura A.3. Formato de opciones de encabezado “*Hop-by-Hop*” y “*Destination*”.

Este campo *Options* contiene opciones de longitud variable en el formato que se ilustra en la figura 1.4, comúnmente conocido como formato *Type Length Value* (TLV), donde el campo *Type* tiene el formato mostrado en la figura 1.5.

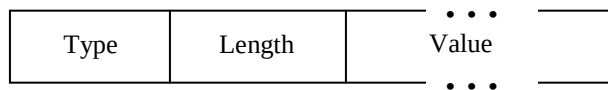


Figura A.4. Formato del campo *Options*.

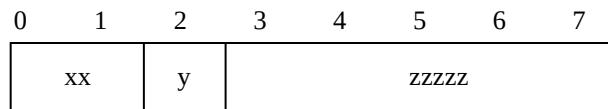


Figura A.5. Formato del campo *Type* del TLV en IPv6.

- **xx** es un número de 2 bits que indica como un nodo que no reconoce la opción deberá tratarla.
- 0 Salta la opción y continúa.
 - 1 Descarta el paquete sin informar.
 - 2 Descarta el paquete y envía un mensaje ICMP *Unrecognized*.
 - 3 Descarta el paquete y envía un mensaje ICMP *Unrecognized* a menos que la dirección destino sea una dirección *multicast*.

- **y** si esta puesto este bit, indica que el valor de la opción puede cambiar en la ruta.
- **zzzzz** los bits restantes defines la opción:
 - 0 Rellena 1 bite de “0”.
 - 1 Rellena n bits de “0”.
 - 194 Longitud de una carga *Jumbo* (cuando el tamaño del paquete excede de 65,535 bytes).

El campo *Length* es el valor del campo *Value* en bytes, y el campo *Value* es dependiente del tipo.

La opción de encabezado *Destination*.

Este encabezado contiene dos campos más opciones, como se muestra en la figura A.3. El campo *Next Header* es de 8 bits e identifica el encabezado inmediatamente siguiente al encabezado *Destination*. El campo *Header Extension Length* es de 8 bits y mide la longitud del encabezado *Destination* en unidades de 8 bytes, sin contar los primeros 8 bytes. El campo *Options* es de longitud variable para completar el encabezado de tal forma que su longitud sea un múltiplo de 8 bytes.

Cuando este encabezado se presenta junto con el encabezado de *Routing* es procesado por el primer destino que aparece en el campo IPv6 de dirección destino, más los subsecuentes destinos listados en el encabezado de *Routing*.

Cuando este encabezado se presenta justo antes del encabezado de la capa superior, esta opción es procesada solo por el destino final del paquete. Esta opción es utilizada solamente para llevar información opcional que necesita ser examinada solo por el nodo destino.

El encabezado *Routing*.

Este encabezado lista uno o más nodos intermedios que son visitados en la trayectoria de la fuente al destino. En la figura A.6 se muestran los campos de este encabezado.

El campo *Next Header* es de 8 bits e identifica el encabezado inmediatamente siguiente a este encabezado de *Routing* y el campo *Header Ext Length* también de 8 bits, mide la longitud del encabezado de *Routing* en unidades de 8 bytes sin contar los primeros 8 bytes.

El campo *RoutingType* de 8 bits, identifica una variante del encabezado de ruteo particular.

El campo *Segments Left* de 8 bits, indica el número de segmentos de ruta restantes. El campo *Type-Specific Data* es de longitud variable con un formato definido por la variante particular de tipo de ruteo.

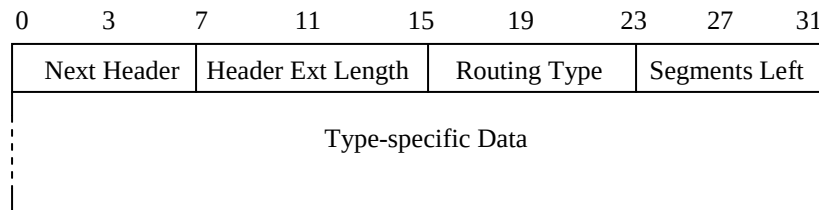


Figura A.6. Formato del encabezado “*Routing*”.

Hasta la fecha [DeS+98] solo se ha definido una variante ilustrada en la figura A.7, el *encabezado de ruteo tipo 0*, el cual contiene una lista ordenada de direcciones que son visitadas junto con el paquete.

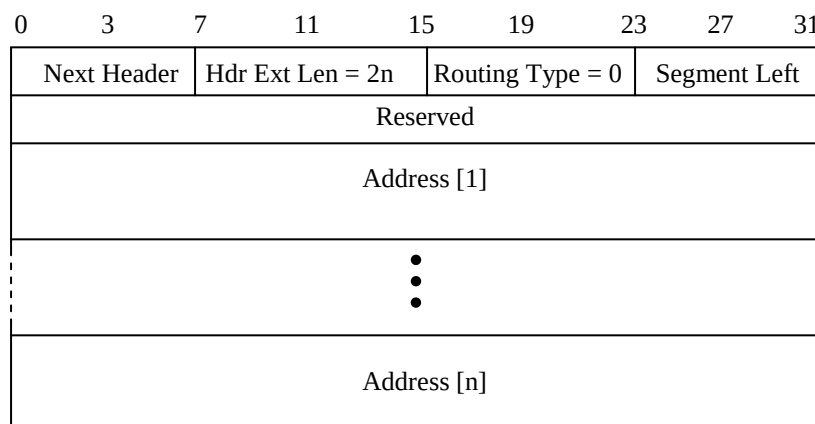


Figura A.7. Formato del encabezado de *Routing* (tipo 0)

Es importante notar que el campo *Header Ext Length* en la figura A.7 contiene un número igual a dos veces el número de direcciones en el encabezado.

El encabezado *Fragment*.

Es utilizada por el emisor para enviar paquetes que son más grandes que la MTU a su destino [MoJ+90].

Un aspecto bastante importante diseñado para este encabezado, es que la fragmentación para IPv6 es solamente hecha por el nodo fuente, no por los ruteadores

intermedios por los que atraviesa la trayectoria del paquete enviado. Este es un cambio significativo de IPv4.

El encabezado *Fragment* contiene seis campos, ver figura 1.8. El campo *Next Header* es de 8 bits e identifica el encabezado inmediatamente siguiente a este encabezado, el campo *Reserved* de 8 bits esta reservado para uso futuro.

El campo *Fragment offset* de 13 bits mide el desplazamiento en unidades de 8 bytes de los datos siguientes a este encabezado, relativos a el inicio de la parte fragmentable del paquete original.

El campo *Res* de 2 bits, está reservado para uso futuro y la bandera *M* de un bit determina si hay mas fragmentos en camino ($M = 1$) ó es el último fragmento ($M = 0$).

El campo *Identification* de 32 bits es un identificador único del paquete fragmentado. Este campo es generado por el nodo fuente.

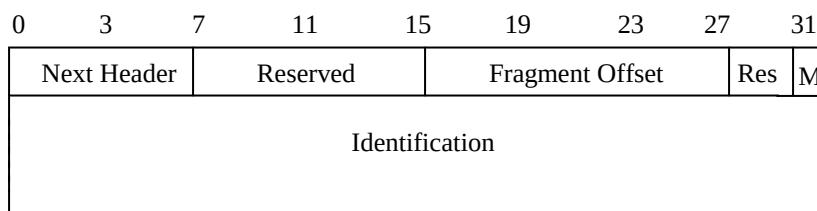


Figura A.8. Formato del encabezado “*Fragment*”.

Cuando es necesario fragmentar un paquete, este es tratado como dos partes, la parte fragmentable y la no fragmentable.

La parte no fragmentable consiste del encabezado más cualquier extensión de encabezado que deba ser procesado en la ruta hacia el destino.

La parte fragmentable es el balance del paquete, el cual puede incluir cualquier extensión de encabezado que es procesado al final del nodo destino, la capa superior y la capa de aplicación de datos.

La parte fragmentable del paquete original, es dividida en paquetes que son múltiplos de ocho bytes, excepto para el último fragmento, el cual puede tener cualquier longitud menor que la MTU.

En la figura A.9-a se muestra el paquete original que requiere fragmentación y en la figura A.9-b se ilustra el paquete ya fragmentado.

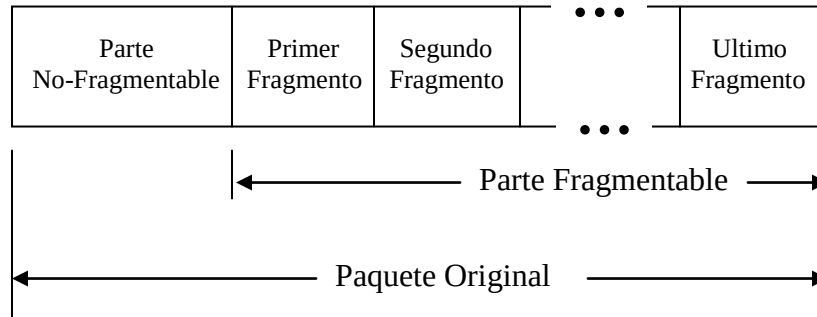


Figura A.9-a. Fragmentación requerida por el paquete original.

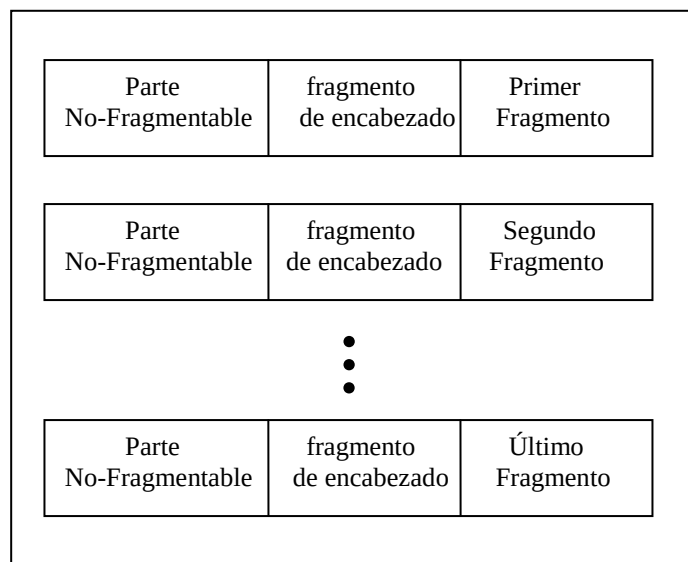


Figura A.9-b. Paquetes fragmentados.

En el nodo destino, el proceso de reensamble es utilizado para reconstruir el paquete original de los paquetes fragmentados.

El encabezado *Authentication*.

Provee integridad sin conexión y autenticación de los datos de origen para los datagramas IP, más protección opcional contra el reenvío.

Este encabezado contiene seis campos como se ilustra en la figura A.10. El campo *Next Header* de 8 bits identifica el encabezado inmediatamente siguiente a este encabezado.

El campo *Payload Length* de 8 bits proporciona la longitud del campo de *Authentication* en palabras de 32 bits menos 2.

El campo *Reserved* de 16 bits es reservado para uso futuro.

El campo *Security Parameters Index* (SPI) de 32 bits identifica la *Security Association* (SA) para este datagrama, relativa a la dirección IP destino contenida en el encabezado IP con la cual este encabezado está asociado y relativa al protocolo de seguridad empleado.

Esta SA es definida [KeS+98b] como una simple conexión lógica que es creada para propósitos de seguridad.

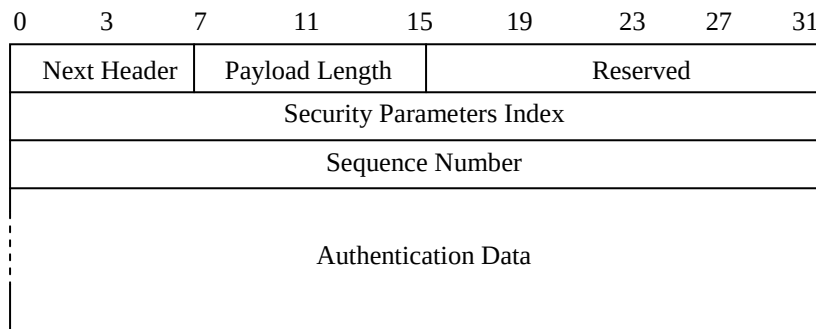


Figura A.10. Formato del encabezado *Authentication*.

El campo *Sequence Number* contiene un número de 32 bits, el cual es incrementado unitariamente. Cuando una SA es establecida, el contador en la fuente y en el destino es inicializado a cero.

El campo *Authentication Data* es de longitud variable y contiene el valor de verificación de integridad *Integrity Check Value* (ICV) para este paquete y debe ser un múltiplo de 32 bits.

El encabezado *Encapsulating Security Payload* (ESP).

Este encabezado fue diseñado para proveer confidencialidad, autenticación de los datos de origen, integridad sin conexión, servicio *anti-replay* y confidencialidad de flujo de tráfico limitado. Los servicios provistos dependen de la SA y de su implementación.

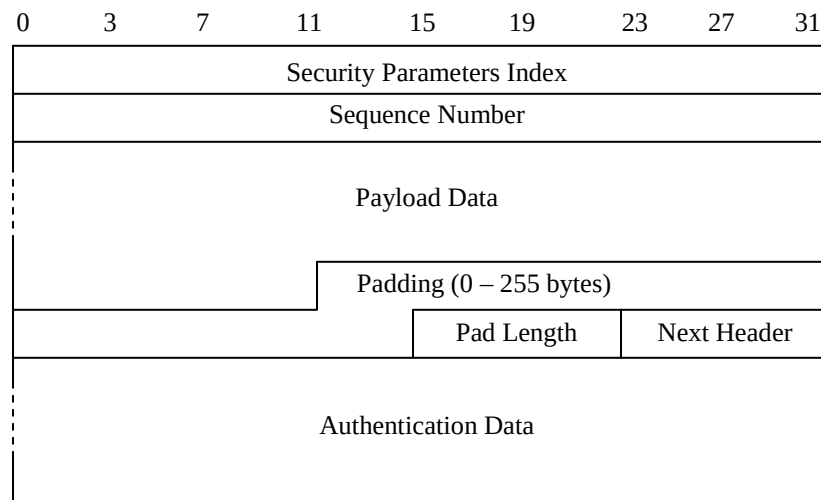


Figura A.11. Formato del encabezado *Encapsulating Security Payload*.

El encabezado contiene 7 campos como es ilustra en la figura A.11, los cuales son:

- El campo *Security Parameters Index (SPI)* de 32 bits identifica la SA para este datagrama, relativa a la dirección destino IP contenida en el encabezado IP con la cual este encabezado de seguridad está asociado y es relativo al protocolo de seguridad empleado.
- El campo *Sequence Number* contiene un número de 32 bits el cual es incrementado unitariamente. Cuando una SA es establecida, el contador en la fuente y en el destino es inicializado a cero.
- El campo *Payload Data* de longitud variable contiene datos descritos por el campo *Next Header*.
- El campo *Padding* opcional de longitud variable contiene información de relleno cuando sea necesaria debido a la implementación de seguridad.
- El campo *Pad Length* de 8 bits indica el número de bytes de relleno del *Padding*.
- El campo *Next Header* identifica el encabezado inmediatamente siguiente al encabezado ESP.
- El campo *Authentication Data* de longitud variable contiene un valor de verificación de integridad (IVC), la longitud de este campo depende de la función de autenticación que es seleccionada. Este campo es opcional y es incluido solo si la SA ha seleccionado el servicio de autenticación.

Apéndice B. Fragmentación y *Tunneling*.

B.1 Fragmentación.

Existen algunas propuestas interesantes a considerar referentes a la fragmentación, algunas de éstas, las describimos a continuación.

La siguiente técnica [MoJ+90] considera que la fragmentación dentro de un túnel puede ser reducida a un mínimo, teniendo el nodo encapsulante datos de la MTU en la trayectoria IPv4 sobre el túnel, utilizando el protocolo de descubrimiento de la MTU y guardando la trayectoria MTU resultante. Así la capa IPv6 en el nodo encapsulante puede entonces ver el túnel como una *capa de enlace de datos* con una MTU igual a la trayectoria IPv4 menos el tamaño del encabezado IPv4 encapsulante.

Esta técnica no elimina completamente la fragmentación. El nodo encapsulante deberá emplear algún algoritmo para determinar cuando transmitir un paquete IPv6 que es mayor que la trayectoria MTU del túnel utilizando la fragmentación IPv4 y cuando retornar un mensaje ICMPv6 de *packet too big*.

Las características comunes con otros tipos de túneles existentes, son las siguientes:

- El nodo de entrada del túnel (nodo encapsulante) añade una cabecera IPv4 y transmite el paquete encapsulado.
- El nodo de salida del túnel (nodo desencapsulante) recibe el paquete encapsulado, quita la cabecera IPv4, actualiza la cabecera IPv6 y procesa el paquete IPv6 recibido.
- El nodo encapsulante puede necesitar mantener información breve para cada túnel registrando parámetros como la MTU del túnel con objeto de procesar los paquetes IPv6 y transmitirlos en el túnel.

El nodo encapsulante que tiene un mayor número de túneles puede no ser capaz de almacenar la trayectoria MTU de IPv4 para todos los túneles. Tales nodos a expensas de fragmentación adicional en la red pueden evitar utilizar el algoritmo mencionado anteriormente empleado sobre el túnel.

Si un paquete tiene un encabezado *fragment*, este indica que el emisor puede o no estar utilizando la trayectoria MTU descubierta por algún medio, es decir, el paquete no tiene el campo de la bandera *DF* habilitada la cual deberá ser trasladada a IPv4.

Si la bandera *DF* no esta habilitada y el paquete IPv4 resulta en un paquete IPv6 (al desencapsularlo) mayor de 1280 bytes, el paquete IPv4 debe ser fragmentado antes de su traslación, ya que paquetes con *DF* deshabilitada, siempre resultarán con un encabezado *fragment* que será agregado a el paquete y estos deberán ser fragmentados de tal forma

que su longitud excluyendo el encabezado IPv4 es cuando más 12332 bytes, es decir, 1280 bytes menos 40 bytes (estos del encabezado IPv6) menos 8 bytes (estos del encabezado *fragment*). De esta forma, los fragmentos resultantes serán trasladados independientemente utilizando esta lógica descrita.

Si el bit *DF* esta habilitado y el paquete no es fragmentado, es decir, la bandera *MF* esta deshabilitada y no el campo *Fragment Offset* es cero, entonces no existe la necesidad de agregar el encabezado *fragment* a el paquete.

Existen algunas diferencias entre IPv4 e IPv6 en cuanto a fragmentación se refiere y el mínimo enlace MTU que afecta el encapsulamiento. En el capítulo 4.4 se menciona el efecto de los distintos valores que un MTU toma en las distintas redes. Ahí se observo que es imposible descubrir un MTU en un enlace emisor-receptor cuando se realiza el encapsulamiento que incluya un trasladador *IPv6-to-IPv4*.

Una solución propuesta [NoE00] a este conflicto, implica que cuando el nodo emisor reciba un mensaje ICMP *packet too big* originado por un ruteador IPv4 que reporte una MTU de 1280, el nodo IPv6 maneje tal mensaje ICMP reduciendo la MTU de la trayectoria a 1280 e incluyendo un encabezado *fragment* con cada paquete. Esto permite que la trayectoria MTU *punto a punto* descubierta sobre el trasladador sea de 1280 o mayor. Ahora, cuando el tamaño del paquete cae por debajo de los 1280, el emisor IPv6 origina un paquete de 1280 bytes que será fragmentado por los ruteadores IPv4 a lo largo de la trayectoria después de que sea trasladado a IPv4.

La única desventaja de este esquema, es que es imposible utilizar la trayectoria MTU *punto a punto* descubierta para optimizar la fragmentación en el emisor, ya que la presencia de un encabezado *fragment* es interpretada como la necesidad de fragmentar el paquete en el lado IPv4. Así, si la aplicación quiere enviar paquetes más grandes independientemente de la trayectoria MTU, el emisor solo será capaz de determinar la MTU en el lado IPv6 del trasladador, y si la trayectoria MTU del lado IPv4 es más pequeña, entonces el emisor IPv6 no recibirá mensajes de error ICMP *packet too big* y no podrá ajustar el tamaño de los fragmentos que envía.

Existen otras reglas para manejar fragmentos y el descubrimiento de la MTU, en donde el encapsulamiento del paquete consiste de un simple mapeo como se describe más adelante. Nótese que los paquetes ICMP requieren un manejo especial, con objeto de trasladar el contenido del mensaje de error ICMP y también de agregarle el *checksum*.

Si el paquete contiene un encabezado *fragment*, los campos del encabezado son definidos como sigue con las siguientes excepciones [NoE00]:

- *Total Length*:
 - Es el valor del campo *Payload length* del encabezado IPv6 menos 8 para el encabezado *fragment*, más el tamaño del encabezado IPv4.
- *Identification*:

- Es copiado de los 16 bits de más bajo orden en el campo *Identification* del encabezado *fragment*.
- *Flags:*
 - La bandera *MF* es copiada de la bandera *M* en el encabezado *fragment*. La bandera del campo *Don't Fragments* es puesta a cero, con el fin de permitir que este paquete sea fragmentado por los ruteadores IPv4.
- *Fragment Offset:*
 - Este campo es copiado del campo *Fragment Offset* en el encabezado *fragment*.
- *Protocol:*
 - El valor del encabezado *Next Header* es copiado del encabezado *fragment*.

Los mecanismos descritos anteriormente nos presentan alternativas para realizar la fragmentación entre redes con distintos protocolos, existen algunos otros mecanismos, y describimos a estas propuestas por considerarlas más interesantes para resolver el problema del descubrimiento del MTU.

B.2 Tunneling.

Los mecanismos de túnel presentados en este documento, suponen que los enlaces son unidireccionales, aún más, los protocolos de las capas superiores manejan los enlaces sobre este hecho. Este tipo de *tunneling* permite a un conjunto de nodos (emisores y receptores) que están directamente conectados por enlaces unidireccionales, enviar paquetes como si estuvieran conectados por un enlace bidireccional.

Otra característica de los métodos de *tunneling* descritos es este documento, es que estos trabajan en la capa tres del modelo OSI.

Existe otro tipo de túneles que trabajan en la capa dos del modelo OSI, y son implementados en la interface de cada nodo conectado a un enlace unidireccional. El objetivo es para ocultar de las capas superiores (capa de red y transporte) la naturaleza unidireccional del enlace [DuE01] y emular la conectividad completamente bidireccional entre cada uno de los nodos.

La desventaja de este tipo de nodos, es que estos no fueron diseñados para topologías donde un par de nodos están conectados por dos enlaces unidireccionales en dirección opuesta, debido a que estos mecanismos requieren que todos los nodos tengan una interface adicional a la infraestructura IP interconectada.

Estos mecanismos de *tunneling* también incluyen un protocolo de configuración de túnel automático que permite a los nodos lograr los enlaces de manera similar a como lo hacen los mecanismos de *tunneling* que trabajan en la capa tres del modelo OSI.

El método de encapsulamiento *GRE* (*Generic Routing Encapsulation*) es sugerido para este tipo de enlaces, el cual proporciona un medio para llevar datagramas IP, ARP y cualquier otro protocolo de la capa tres entre nodos.

Como en otros tipos de túneles, el emisor y receptor deben conocer los puntos de entrada y salida con objeto de transmitir los datagramas encapsulados.

Debido a la naturaleza propia de este mecanismo, el número de emisores y receptores se espera que sea relativamente pequeño, de tal forma que cada nodo emisor tenga una lista de todos los nodos emisores y receptores encapsulantes y desencapsulantes respectivamente. Así, el administrador deberá configurar los túneles para todos los nodos existentes que encapsulan.

En el caso de que un nuevo nodo sea habilitado, cada receptor deberá crear un túnel para habilitar la comunicación bidireccional con este.

Si un nodo es dado de baja o deshabilitado, los receptores deben deshabilitar sus túneles hacia ese nodo. Lo cual previene tráfico innecesario de datagramas siendo encapsulados y así mismo evitar sobrecarga en la red.

El protocolo *DTCP* (*Dynamic Tunnel Configuration Protocol*) provee un medio para que los receptores dinámicamente descubran la presencia de emisores encapsulante y mantengan una lista de los nodos operacionalmente activos. Además, un emisor continuamente deberá anunciar las direcciones del punto de salida del túnel sobre los enlaces, y los receptores deberán escuchar los anuncios para mantener actualizada la lista de los puntos de salida de los túneles activos.

Apéndice C. Pseudo códigos.

```
Packet structure {  
    Heading structure;  
    message[size of the message];  
}packet;  
  
checksum(length, *buffer) {  
    add all the 16 bits words;  
    if a carrying bit exists  
        add to the sum;  
    add to the carrying;  
    add again to the carrying;  
    evaluate the complement to one;  
    return to a value of 16 bits  
}  
  
main{  
    structure definitions {  
        Heading structure;  
    }  
  
    definitions;  
    create the socket raw  
    link the socket  
    connect  
    read information (IPv6 packet)  
    header IPv4 → field of header IPv6  
    define heading options  
    add IPv4 heading  
    send information to destination  
    close socket  
}
```

Figura C1. Pseudo código que agrega un encabezado.

```

main{
  structure definitions {
    Heading structure;
  }
  definitions;
  create the socket raw
  link the socket
  listen
  accept
  read information (encapsulated packet)
  process application
    if the destination is different from the hostname
      send the packet from destination
    else
      eliminate heading
  close socket from the emission side
}

```

Figura C2. Pseudo código que elimina un encabezado.

```

struct ip_packet {
  struct {definitions} header; /* encabezado */
};
void dump(void* b, int len)
{ definitions;
  for ( 0 to paquet_size)
  {
    if ( column = 16 )
    {
      printf(bin value to paquet);
      memset(str, 0, 17);
    }
  }
}
int main()
{ definitions;
  create a socket_raw;
  {
    bytes_read of the paquet;
  }
  while ( bytes_read > 0 );
  return 0;
}

```

Figura C3. Pseudo código de la herramienta “escucha paquetes”.

Bibliografia.

- [Bla+98] Nichols, K., Blake, S., Baker, F. and D. Black, Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers (RFC 2474, December 1998)
- [BoJ+96] J. Bound, B. Carpenter, D. Harrington, J. Houldsworth, A. Lloyd, *OSI NSAPs and IPv6* (RFC 1888, August 1996)
- [BrS+93] S. Bradner + A. Mankin, *IP: Next Generation (IPng) White Paper Solicitation* (RFC 1550, December 1993)
- [BrS+95] S. Bradner, A. Mankin, *The Recommendation for the IP Next Generation Protocol* (RFC 1752, January 1995)
- [DeS+95] S. Deering, R. Hinden *Internet Protocol, Version 6 (IPv6) Specification* (RFC 1883, December 1995)
- [DeS+98] S. Deering, R. Hinden, *Internet Protocol, Version 6 (IPv6) Specification* (RFC 2460, December 1998)
- [DuE01] E. Duros, W. Dabbous, H. Izumiyama, N. Fujii, Y. Zhang. *A Link-Layer Tunneling Mechanism for Unidirectional Links* (RFC 3077 March 2001)
- [FuV+93] V. Fuller, T. Li, J. Yu, K. Varadhan, Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy (RFC 1519, September 1993.)
- [GiR+96] R. Gilligan, E. Nordmark, *Transition Mechanisms for IPv6 Hosts and Routers* (RFC 1933, April 1996)
- [HaS+94] S. Hanks, T. Li and P. Trania, *Generic Routing Encapsulation (GRE)* (RFC 1701, October 1994)
- [HeC88] C. L. Hedrick, *Routing Information Protocol* (RFC 1058, Jun-01-1988)
- [HiR+98a] R. Hinden, S. Deering, *IP Version 6 Addressing Architecture* (RFC 2373, July 1998)
- [HiR+98b] R. Hinden, M. O'Dell, S. Deering, *An IPv6 Aggregatable Global Unicast Address Format* (RFC 2374, July 1998)
- [KeS+98a] S. Kent, R. Atkinson, *IP Encapsulating Security Payload (ESP)* (RFC 2406, November 1998)

- [KeS+98b] S. Kent, R. Atkinson, *Security Architecture for the Internet Protocol* (RFC 2401, November 1998)
- [MaG+97] G. Mal, R. Minnear, *RIPng for IPv6* (RFC 2080, January 1997)
- [MiM99] Mark A. Miller P. E., *Implementing IPv6*, (Foster City, CA: M&T Books, 1999)
- [MoJ+90] J.C. Mogul, S.E. Deering, *Path MTU discovery* (RFC 1191, Nov-01-1990)
- [NaT+98] T. Narten, E. Nordmark, W. Simpson, *Neighbor Discovery for IP Version 6 (IPv6)* (RFC 2461 December 1988)
- [NoE00] E. Nordmark, *Stateless IP/ICMP Translation Algorithm (SIIT)* (RFC 2765 February 2000)
- [PaC+94] C. Partridge, F. Kastenholz, *Technical Criteria for Choosing IP The Next Generation (IPng)* (RFC 1726, December 1994)
- [PaC95] C. Partridge, *Using the Flow Label Field in IPv6* (RFC 1809, June 1995)
- [PoJ81] J. Postel , *Internet Protocol* (RFC 791, Sep-01-1981)
- [ReY+93a] Y. Rekhter, T. Li, *An Architecture for IP Address Allocation with CIDR* (RFC 1518, September 1993).
- [ReY+93b] Y. Rekhter, C. Topolcic, *Exchanging Routing Information across Provider Boundaries in the CIDR Environment* (RFC 1520, September 1993)
- [SUN99] Sun Microsystems Team, *IPv6 and the future of internet. A technical white paper* (Sun Microsystems, November 1999)
- [ThS+95] S. Thomson, C. Huitema, *DNS Extensions to support IP version 6* (RFC 1886, December 1995)
- [WaS01] Sean Walton, *Programación de Socket Linux* (Madrid España: Prentice Hall, 2001).
- [LoM+03] M. Oswaldo López, Jesús García F. *IPv6 Encapsulating Techniques* (Cancún Mex: ACTA Press, ISBN: 0-88986-3490, 2003)