DISEÑO E IMPLEMENTACIÓN DE UN LABORATORIO INTERNET IPV6 DESDE LA PERSPECTIVA DE ENRUTAMIENTO, SEGURIDAD Y QOS

Liliana Andrea Urbano Ramos
Diego Alejandro Paz Cabrera

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telecomunicaciones
Popayán
2004

DISEÑO E IMPLEMENTACIÓN DE UN LABORATORIO INTERNET IPV6 DESDE LA PERSPECTIVA DE ENRUTAMIENTO, SEGURIDAD Y QOS

Liliana Andrea Urbano Ramos
Diego Alejandro Paz Cabrera

Documento Final de Trabajo de Grado

Director: Ing. Francisco Javier Teran

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telecomunicaciones
Popayán
2004

TABLA DE CONTENIDO

1 Introducción

Una de las características que mejor define a las sociedades desarrolladas en la actualidad es la creciente utilización de las nuevas tecnologías de la información y las comunicaciones. En este contexto Internet - *la red de redes* - ha reevaluado la posición del hombre frente al conocimiento, estableciendo nuevos paradigmas que se caracterizan por la infinidad de posibilidades que las autopistas de información aportan a la vida de las personas.

Todo esto no sólo ha hecho que Internet - *basada en un diseño de comienzos de los años 80* -, haya experimentado un crecimiento sin comparación en la historia de las telecomunicaciones, sino que también ha provocado que las aplicaciones basadas en el protocolo IP requieran una serie de servicios tales como el soporte de tráfico en tiempo real, esquemas flexibles de control de la congestión y seguridad a nivel de red, difíciles de imaginar cuando se desarrolló IPv4. Por tal razón para remediar estos problemas, los cuerpos técnicos de Internet impulsaron un debate bajo el lema IPng[1] que culminó con la especificación de un nuevo protocolo IP, sucesor del actual IPv4, conocido formalmente como la versión 6 del Protocolo Internet o **IPv6**.

La evolución gradual de IPv6 hacia una nueva visión global de Internet caracterizada por mayores niveles de seguridad, autoconfiguración de enrutadores y equipos terminales de usuario, movilidad, QoS (Quality of Service: *Calidad de Servicio*), transporte de tráfico multimedia en tiempo real, aplicaciones “Anycast” y “Multicast”, transición de IPv4 a IPv6 y lo que es más importante, mayor espacio de direcciones; está provocando que fabricantes, organismos de investigación y desarrollo, instituciones académicas, operadores de telecomunicaciones y líderes proveedores de soluciones y servicios de Internet, orienten sus esfuerzos en la creación de la próxima generación de Internet.

Es así como este Laboratorio Internet IPv6, establece un polo de desarrollo tecnológico en el área de IPv6, siendo el punto de partida en la implementación práctica de esta nueva tecnología en la red de datos de la Universidad del Cauca, a través de la creación, operación y administración de la Red piloto Unicauca IPv6 y su integración en 6Bone - *la red internacional experimental de IPv6 sobre Internet* - mediante la asignación del espacio de direcciones IPv6 (3FFE:8070:1024::/48 - bloque pNLA /48) a partir del prefijo tipo pTLA (3FFE:8070::/28) que tiene asignado la Universidad Nacional Autónoma de México.

Tres aspectos significativos conforman el escenario objetivo de éste Laboratorio: Enrutamiento, Seguridad y QoS en redes IPv6, los cuales facilitan una aproximación directa a IPv6 que contribuye a consolidar en la comunidad estudiantil el uso y conocimiento de ésta tecnología y sus aplicaciones para migrar la red UniCauca IPv4 hacia una red de nueva generación.

[1] IP Next Generation: Protocolo de Internet de Próxima Generación

2 Fundamentos de la Tecnología IPv6

2.1 La necesidad de un nuevo protocolo IP: IPng

Hacia 1990, las previsiones de que en un futuro no muy lejano las direcciones IP podrían agotarse llevaron al *Internet Engineering Task Force*, Grupo de trabajo de ingeniería en Internet (IETF) a iniciar una serie de acciones encaminadas a obtener un nuevo protocolo IP. Si bien los 32bits en la estructura de las direcciones IPv4 permiten enumerar 4,294,967,296 posibles direcciones, la realidad está muy alejada de estas cifras. La estructura inicial utilizada para la asignación de direcciones con dos niveles - *segmento de red, segmento de host* - desaprovecha una gran cantidad del espacio de direccionamiento. Esta ineficiencia se ha visto incrementada por la organización en el reparto de rangos de direcciones en las Clases A, B, C y D[2].

Ante esta situación en Noviembre de 1991 el IETF formó el grupo de trabajo *ROuting and ADdressing* (ROAD), destinado a estudiar los problemas de enrutamiento y direccionamiento asociados al crecimiento de Internet. Este grupo hizo una serie de propuestas que abarcaban desde soluciones a corto plazo, como el *Classless Inter-Domain Routing*, Enrutamiento entre dominios sin clase (CIDR), hasta la recomendación de una solicitud de propuestas para un nuevo protocolo IP con direcciones de mayor tamaño. Dicha solicitud de propuestas para un protocolo de Próxima Generación se llevó a cabo en Julio de 1992 y quedó recogida en el RFC 1550[3]. Se recibieron veintiún propuestas, algunas de ellas procedentes de industrias que se esperaba se convirtieran en importantes mercados para las redes de datos: la TV por cable, la industria de la telefonía móvil y la industria eléctrica. Todas las propuestas recibidas fueron examinadas por el área IPng, grupo creado al efecto por el IETF.

2.2 Criterios técnicos para IPng

Como resultado de los trabajos del área IPng se obtuvo el documento de “Criterios técnicos para IPng”. En él se recogieron los requisitos que debía cumplir el nuevo protocolo, así como los criterios para hacer una recomendación específica sobre IPng:

[2] División del rango de direcciones IPv4

Clase	Bits más significativos	Bits Segmento de Red	No. de redes	Bits Segmento de Host	No. de Hosts
A	0	7	124	24	16,777,214
B	01	14	16,382	16	65,534
C	110	21	2,097,152	8	254
D	Las direcciones de clase D son un grupo especial que se utiliza para dirigirse a grupos de hosts. Estas direcciones son muy poco utilizadas. Los cuatro primeros bits de una dirección de clase D son 1110				

[3] IP Next Generation (IPng) White Paper Solicitation

- **Escalabilidad:** El nuevo protocolo debe permitir direccionar al menos 10^9 redes. Se debe prestar especial atención a cómo se solucionan los problemas de direccionamiento y de agregación de rutas.

- **Flexibilidad topológica:** La arquitectura de enrutamiento y los protocolos asociados a IPng han de permitir cualquier tipo de topología.

- **Rendimiento:** El nuevo protocolo no debe suponer una penalización en el tiempo de tratamiento del tráfico IP por parte de los enrutadores, de manera que éstos permitan el adecuado aprovechamiento de la creciente velocidad de los diferentes medios físicos de transmisión.

- **Servicio robusto:** El servicio de red y los protocolos de enrutamiento y control asociados deben ser al menos tan robustos como los de IPv4 en cuanto al manejo de paquetes defectuosos, fallos de enlaces y dispositivos de red, ataques malintencionados ...

- **Transición:** El nuevo protocolo debe contar con un plan de transición desde el actual IPv4, que permita la coexistencia de ambos durante el tiempo que sea necesario.

- **Independencia del medio:** El nuevo protocolo debe ser capaz de trabajar tanto en redes con velocidades del orden de varios cientos de Gigabits por segundo, como en redes de muy baja velocidad.

- **Seguridad:** La introducción de gran público en Internet ha propiciado la aparición de nuevas aplicaciones, como el comercio electrónico, que requieren un nivel de red que proporcione seguridad.

- **Extensibilidad:** Para que no le suceda igual que a IPv4, el nuevo protocolo debe estar diseñado de forma que permita evolucionar y acomodarse a futuros servicios que Internet pueda demandar.

- **Clases de servicio:** IP es un protocolo de entrega orientado al mejor esfuerzo “best-effort”, pero cierto tipo de tráfico, como audio y vídeo, requieren un cierto control. Los dispositivos de red (hosts y enrutadores) deben ser capaces de asociar cada paquete con una clase de servicio específico, asegurando a dichos paquetes los servicios asociados a esa clase.

- **Movilidad:** El nuevo protocolo debe soportar nodos fijos y móviles. Movilidad entendida desde dos perspectivas: por un lado, nodos móviles que mantienen conexiones a través de enlaces inalámbricos y por otro, nodos que se desconectan de un punto de la red para conectarse en otro diferente y seguir trabajando desde allí.

Además el nuevo protocolo debe mantener algunos aspectos de IPv4, y mejorar otros, por ejemplo, el soporte Multicast. También se busca facilitar la configuración, administración y operación de los dispositivos.

2.3 Propuestas para IPng

Entre las propuestas que fueron surgiendo a lo largo de todo el proceso, cabe destacar las siguientes:

- **TUBA - TCP and UDP with Bigger Addresses**

Consiste, a grandes rasgos, en reemplazar IP por el *Connection-Less Network Protocol*, Protocolo de redes sin conexión (CLNP), lo que permitiría usar direcciones más grandes, manteniendo el resto de los protocolos. TCP, UDP y las aplicaciones tradicionales de TCP/IP estarían por encima de CLNP.

- **CATNIP - Common Architecture for the Internet**

Fue concebido como un protocolo de convergencia, que integraba CLNP, IP e IPX, de forma que dos hosts con diferentes niveles de red, por ejemplo IP y CLNP, pudieran comunicarse sin problemas. Utilizaba direcciones de 64bits.

- **SIPP - Simple Internet Protocol Plus**

Estaba caracterizado por direcciones de 64bits y fue diseñado de forma que tuviera una evolución natural desde IPv4. Además de aumentar el rango de direcciones, en él se incluían algunas de las características que eran deseables para el nuevo protocolo:

  - *Simplificación de cabeceras*: Eliminaba algunos campos en la cabecera IPv4 a la vez que reestructuraba otros para simplificar las tareas de enrutamiento.

  - *Mejoras en la implementación de opciones*: Se adicionaron las opciones propias de IPng mediante cabeceras de Extensión.

  - *Calidad de Servicio*: SIPP ofrecía la posibilidad de etiquetar paquetes como pertenecientes a un determinado flujo, y que los hosts demandasen de los enrutadores un determinado trato para ese flujo, una mejora especialmente útil para aplicaciones de audio y vídeo en tiempo real.

  - *Autenticación y privacidad*

Tras estudiar detenidamente las propuestas, si bien se concluyó que tanto TUBA como SIPP podían ser opciones válidas, también se reconoció que cada uno de ellas tiene sus propias deficiencias. El proceso de discusión entorno a IPng continuó y finalmente el protocolo descrito en el documento "Simple Internet Protocol Plus (SIPP)" se seleccionó como la base para IPng. Se trata del protocolo SIPP mejorado con direcciones de 128bits.

Asimismo, otros protocolos propuestos contribuyeron en aquellas áreas en las que SIPP presentaba mayores carencias, como la definición de un escenario de transición o la posibilidad de autoconfiguración, campos en los que se optó por las propuestas de TUBA.

2.4 Características de IPv6

IPng fue diseñado como una evolución de IPv4, manteniendo aquellas funciones que eran satisfactorias y eliminando aquellas que no lo eran. La primera versión del IP de Próxima Generación, apareció en 1996 bajo el nombre de IPv6. Los principales signos de identidad que le diferencian de su predecesor son:

- **Direccionamiento y Enrutamiento**: El tamaño de las direcciones IP pasa de 32 a 128bits. Esto supone un espacio de direccionamiento 2^{96} veces mayor, lo que posibilita una jerarquía de asignación de más niveles y una autoconfiguración más simple de las direcciones. La escalabilidad del enrutamiento Multicast se ve mejorada con la adición de un campo *scope* (ámbito) a las direcciones Multicast para limitar el ámbito de un grupo Multicast.

- **Formato simplificado de la cabecera:** Algunos campos de la cabecera IPv4 se han eliminado o se han hecho opcionales. De esta forma, se simplifica y agiliza el manejo de los paquetes IPv6 por parte de los enrutadores.

- **Soporte mejorado de opciones:** En IPv6 las opciones se implementan en cabeceras de Extensión separadas. Esta organización supone una gran mejora en cuanto al rendimiento de los enrutadores, puesto que en la mayoría de los casos estas opciones no se procesan hasta que el paquete alcanza su destino final.

- **Extensibilidad:** Es posible indicar las acciones a realizar por el enrutador en caso de que desconozca alguna opción, codificando dicha información dentro de la propia opción. Esta es una característica muy importante para posibilitar el desarrollo incremental de nuevas funcionalidades.

- **Autenticación y privacidad:** IPv6 define dos extensiones relacionadas con este tema: el IP Authentication Header (AH) y el IP Encapsulating Security Payload (ESP). El primero de

ellos garantiza que se ha realizado la transmisión del mensaje sin errores ni modificaciones. El segundo provee de un mecanismo para cifrar la carga útil del paquete o incluso el propio paquete, permitiendo la transmisión de información sensible a salvo de posibles intromisiones.

- **Calidad de servicio:** IPv6 implementa el concepto de flujo, es decir, una secuencia de paquetes entre dos nodos para los cuales el nodo origen desea un tratamiento determinado por parte de los enrutadores intermedios. Esto hace que los enrutadores tengan que llevar el control de los flujos y cierta información sobre cómo tratar el paquete, pero, a cambio, acelera el tiempo de procesamiento pues si no, habría que incluir dicha información en alguna extensión que tendría que ser procesada por los enrutadores.

2.5 Especificaciones Básicas de IPv6

2.5.1 Formato y contenido de la Cabecera de un paquete IPv4

La cabecera de un paquete IPv4 es de longitud variable, y se especifica utilizando múltiplos de 32bits. La longitud mínima - *sin incluir el campo Opciones* - de la cabecera IPv4 es 20bytes[4], pero si se agregan Opciones, la cabecera se extiende en múltiplos de 4bytes hasta 60bytes para transmitir información adicional relacionada con seguridad, enrutamiento de origen, registro de ruta, identificación de flujo, y/o marca de tiempo.

Los campos que definen el formato de la cabecera de un paquete IPv4 se ilustran a continuación:

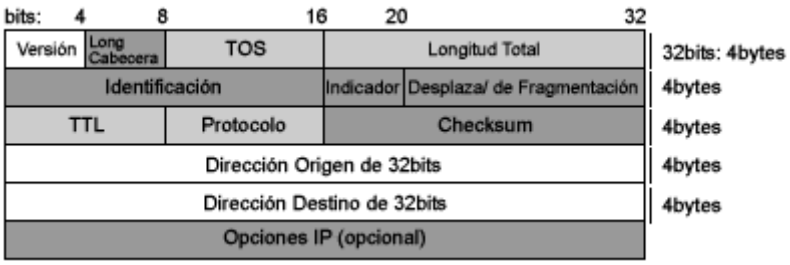


Figura 1 - Descripción de la cabecera de un paquete IPv4

- Versión, *Version* (4bits)
- Longitud de la Cabecera, *Header Length* (4bits)
- Tipo de Servicio, *Type Of Service* “TOS” (1byte)
- Longitud Total, *Total Length* (2bytes)
- Identificación, *Identification* (2bytes)
- Indicador, *Flag* (4bits)

[4] Cada fila en el formato de la Cabecera de un paquete IPv4 contiene 4bytes.

- Desplazamiento de Fragmentación, *Fragment Offset* (12bits)
- Tiempo de vida, *Time To Live* "TTL" (1byte)
- Protocolo, *Protocol* (1byte)
- Código de Verificación, *Checksum* (2bytes)
- Dirección Origen de 32bits (4bytes)
- Dirección Destino de 32bits (4bytes)

En la Figura 1, se distinguen los campos de la Cabecera de un paquete IPv4, que se mantienen, se modifican y/o desaparecen en la nueva versión del protocolo IP, según el siguiente esquema:

Nota: Los campos Dirección Origen y Dirección Destino de 32bits en la cabecera IPv4, se mantienen en la cabecera IPv6, aunque extienden su tamaño a 128bits respectivamente.

2.5.2 **Formato y contenido de la Cabecera de un paquete IPv6**

La estructura de la cabecera de un paquete IPv6 es más simple y más eficiente que la estructura de una cabecera IPv4. La longitud de ésta cabecera tiene un tamaño fijo de 40bytes, y contiene 8 campos a diferencia de la cabecera de un paquete IPv4 que contiene como mínimo 12 campos.

IPv6 ha eliminado 5 campos de la cabecera IPv4, estos son:

- Longitud de la Cabecera
- Identificación
- Indicador
- Desplazamiento de Fragmentación, *Fragment Offset*
- Código de Verificación, *Checksum*

El motivo fundamental por el que los campos son eliminados, es su innecesaria redundancia.

En el caso del campo Longitud de la Cabecera, éste campo es inútil ya que IPv6 define una cabecera con una longitud fija, y si es preciso agregar Opciones, éstas se especifican utilizando cabeceras de Extensión.

Los campos Identificación, Indicador y Desplazamiento de Fragmentación, negocian la fragmentación[5] en paquetes IPv4, ya que un enrutador IPv4 puede fragmentar un paquete original en paquetes constitutivos más pequeños y reenviar múltiples paquetes, para luego, en el destino, recibir los paquetes y reensamblarlos en un único paquete. Si un paquete es perdido o tiene un error, el proceso es ineficiente, ya que la transmisión se inicia nuevamente. En IPv6, los nodos aprenden el tamaño de la Unidad de Transmisión Máxima (MTU), a través de un procedimiento de inspección de ruta llamado "Descubrimiento de MTU". De esta manera, los enrutadores IPv6 a lo largo de una ruta, no realizan fragmentación de paquetes, sino que de ser necesario, ésta operación se efectúa extremo a extremo utilizando cabeceras de Extensión, por lo que son innecesarios los campos Identificación, Indicador y Desplazamiento de Fragmentación.

El campo Código de Verificación de la integridad de la cabecera IPv4, cuya función ya es realizada por otros mecanismos de nivel inferior, como es el caso de la norma IEEE 802.x MAC y de la *ATM adaptation layer*, Capa de adaptación ATM (AAL), se elimina en IPv6, ya que el riesgo de no detectar errores es mínimo y el aumento en la velocidad de procesamiento de los paquetes por parte de los enrutadores IPv6, crece considerablemente. Además, IP es un protocolo de entrega orientado al mejor esfuerzo "best-effort", por lo que es responsabilidad de los protocolos de nivel superior asegurar su integridad.

2.5.3 **Campos en la Cabecera de un paquete IPv6**

Los campos que definen el formato de la cabecera de un paquete IPv6 se ilustran a continuación:

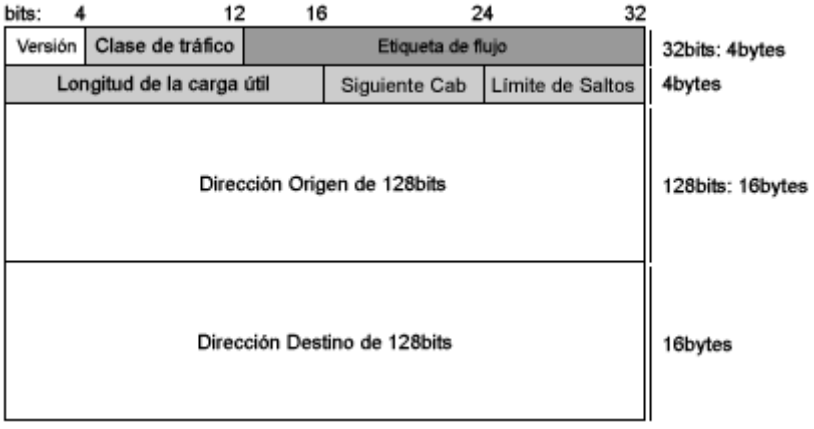
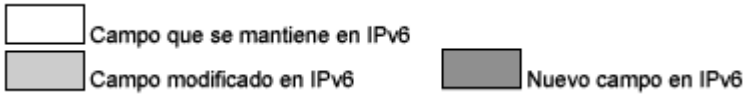


Figura 2 - Descripción de la Cabecera de un paquete IPv6

[5] La fragmentación se da, cuando un paquete se envía sobre redes físicas que tienen un tamaño máximo de trama ó una Unidad de Transmisión Máxima - Maximum Transmission Unit (MTU), lo que no permite que paquetes de mayor tamaño sean colocados en una trama física.

Algunos campos de la cabecera IPv4 se conservan en IPv6, otros se han modificado adoptando nuevas denominaciones, y otros se han adicionado para conformar la cabecera básica de un paquete IPv6, según el siguiente esquema:



Los campos en la cabecera de un paquete IPv6, son los siguientes:

2.5.3.1 **Versión**, *Version* (4bits): Este campo indica la versión del protocolo IP. En el caso de IPv6, el número de versión es el 6. El número de versión 5 no puede ser usado, debido a que ya ha sido asignado a un protocolo experimental (ST2, RFC1819).

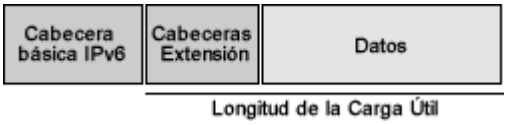
2.5.3.2 **Clase de tráfico**, *Traffic Class* (1byte): Este campo reemplaza al campo Tipo de Servicio en IPv4 y tiene como función, etiquetar paquetes con una clase de tráfico para utilizarse en Servicios Diferenciados.

El RFC 2474, "Definición del Campo de Servicios Diferenciados (DS) en cabeceras IPv4 e IPv6" explica como el campo Clase de tráfico puede ser utilizado.

2.5.3.3 **Etiqueta de flujo**, *Flow Label* (20bits): Este nuevo campo se utiliza para etiquetar paquetes de un flujo específico, y así diferenciarlos de otros paquetes en el nivel de red. Con esta etiqueta, un enrutador no necesita examinar el interior del paquete para identificar sus características, ya que la información esta disponible en la cabecera del paquete IPv6.

Los campos Clase de tráfico y Etiqueta de flujo, son los que permiten una de las características fundamentales e intrínsecas de IPv6: *Quality of Service*, Calidad de Servicio (QoS), a partir de mecanismos de control de flujo y de asignación de prioridades diferenciadas según el tipo de servicio.

2.5.3.4 **Longitud de la carga útil**, *Payload Length* (2bytes): Este campo, nombrado en IPv4 como Longitud Total, especifica la longitud de la carga útil *(payload)*, o la longitud de los datos transportados después de la cabecera básica IPv6. Es de señalar que las cabeceras de Extensión se consideran parte del *payload* IPv6.



Debido a que el campo Longitud de la carga útil, tiene una longitud de 2bytes, se limita el tamaño del *payload* IPv6 a 65,536bytes. Sin embargo, IPv6 dispone de la opción Jumbograma[6], descrita en el RFC 2675, "IPv6 Jumbograms", la cual soporta tamaños grandes de *payload*, si es necesario.

Nota: El cálculo del *payload* en IPv6 es diferente al realizado en IPv4. El campo Longitud Total en IPv4 define la longitud total del paquete, incluidas las partes de cabecera y de datos de usuario.

2.5.3.5 **Siguiente Cabecera**, *Next Header* (1byte): En IPv4, este campo es conocido como Protocolo, y es modificado en IPv6 para transportar información opcional que se codifica en cabeceras de Extensión. Así, el valor del campo Siguiente Cabecera determina el tipo de cabecera de Extensión que complementa la cabecera básica de un paquete IPv6.

2.5.3.6 **Límite de saltos**, *Hop Limit* (1byte): Similar al campo Tiempo de vida en IPv4. El valor del campo Límite de saltos en IPv6 especifica el número máximo de enrutadores que un paquete puede transitar antes de ser descartado. Decrementa en 1 cada vez que un paquete atraviesa un nodo IPv6. Si llega a 0 se descarta.

2.5.3.7 **Dirección Origen**, *Source Address* (16bytes): Este campo contiene la dirección IP de quien origina el paquete IPv6.

2.5.3.8 **Dirección Destino***, Destination Address* (16bytes): Este campo contiene la dirección IP del destino del paquete IPv6.

Nota: En la Figura 2, se observa que los campos Dirección Origen y Dirección Destino emplean un total de 32bytes de los 40bytes con que cuenta la cabecera básica de un paquete IPv6, por lo que únicamente 8bytes se utilizan para administrar la información que maneja la cabecera IPv6.

2.5.4 Cabeceras de Extensión IPv6

IPv6 codifica información opcional en cabeceras de Extensión independientes, localizadas entre la cabecera básica IPv6 y una cabecera TCP o UDP. En consecuencia, IPv6 elimina el campo Opciones de la cabecera IPv4.

[6] Jumbograma es una opción que permite que la longitud máxima de los datos transportados por IPv6 (16bits, 65,536bytes), se extienda hasta 64bits. Se prevé su uso especialmente para tráficos multimedia, sobre líneas de banda ancha. Sin embargo estos paquetes no pueden ser fragmentados.

Nota: Cada cabecera de Extensión incluye su propio campo Siguiente Cabecera, de tal forma que sucesivas cabeceras de Extensión enlazadas, pueden estructurar adecuadamente un paquete IPv6[7].

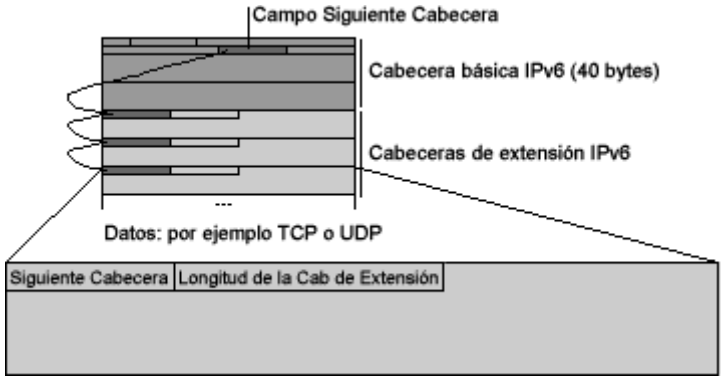


Figura 3 - Cabeceras de Extensión IPv6

A continuación se muestra una lista de posibles valores para el campo Siguiente Cabecera, los cuales establecen diferentes tipos de cabeceras de Extensión que complementan la cabecera básica de un paquete IPv6:

Valor	**Descripción**
0	*Hop-by-Hop option header*, Cabecera de opciones salto a salto
1	ICMPv4 - *Internet control message protocol*, Protocolo de mensajes de control de Internet IPv4
2	IGMPv4 - *Internet group management protocol*, Protocolo de gestión de grupos de Internet IPv4
4	IP en IP (encapsulación)
6	TCP *Transmission control protocol*, Protocolo de control de transmisión
8	EGP - *Exterior gateway protocol*, Protocolo de gateway Exterior
9	IGP - *Interior gateway protocol*, Protocolo de gateway Interior
17	UDP - *User datagram protocol*, Protocolo de datagrama de usuario
41	IPv6 - Protocolo de Internet versión 6
43	*Routing header*, Cabecera de Enrutamiento
44	*Fragmentation header*, Cabecera de Fragmentación
45	IDRP - *Interdomain routing protocol*, Protocolo de enrutamiento entre dominios
50	ESP - Encrypted security payload header
51	AH - Authentication header
58	ICMPv6 - *Internet control message protocol*, Protocolo de mensajes de control de Internet IPv6
60	*Destination options header*, Cabecera de opciones de Destino
88	EIGRP - *Enhanced interior gateway routing protocol*, Protocolo de enrutamiento de gateway interior mejorado
89	OSPF - *Open shortest path first*, Primero la ruta libre más corta
115	L2TP - *Layer 2 tunneling protocol*, Protocolo de túnel de nivel 2
134-254	Sin asignar
255	Reservado

Tabla 1 - Valores en el campo Siguiente Cabecera

Estas cabeceras de Extensión son procesadas en el estricto orden en que aparecen en un paquete IPv6, únicamente por su enrutador IPv6 final, identificado por el campo Dirección

[7] Si paquetes IPv6 se encapsulan en paquetes IPv4, la cabecera de nivel superior puede ser otra cabecera IPv6 que puede contener sus propias cabeceras de Extensión.

destino en la cabecera básica IPv6[8]. Pero existe una excepción, si la cabecera de Extensión es *Hop-by-Hop option header*, cabecera de opciones salto a salto, la información transportada por el paquete IPv6 debe ser examinada por cada uno de los enrutadores a lo largo de la ruta del paquete.

En caso de emplearse más de una cabecera de Extensión en un único paquete, se recomienda que éstas aparezcan de acuerdo al siguiente orden:

- Cabecera básica IPv6.
- Cabecera de opciones salto a salto.
- Cabecera de opciones de Destino: Opciones a ser procesadas por el destino final[9] definido en la cabecera básica IPv6, además de los destinos adicionales listados en la Cabecera de Enrutamiento.
- Cabecera de Enrutamiento.
- Cabecera de Fragmentación.
- Cabecera AH.
- Cabecera ESP.
- Cabecera de opciones de Destino: Opciones a ser procesadas únicamente por el destino final del paquete IPv6.
- Cabecera de nivel Superior.

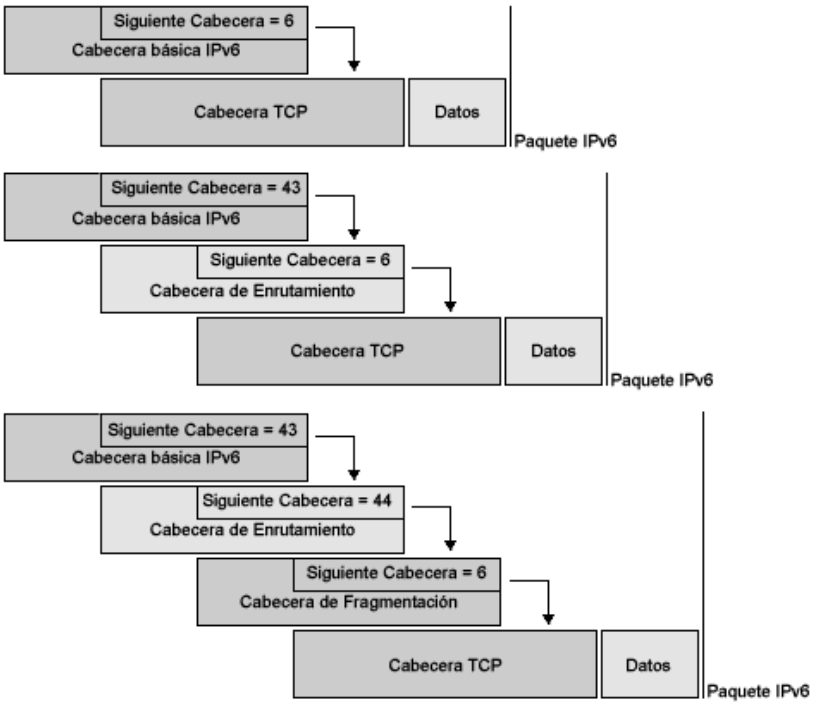


Figura 4 - Uso de sucesivas Cabeceras de Extensión

[8] Todo nodo IPv6 intermedio *no* debe examinar más que la cabecera IPv6 básica de un paquete IPv6. Si la dirección en el campo Dirección destino es Multicast, las cabeceras de Extensión son examinadas por todos los enrutadores IPv6 que pertenecen al grupo Multicast.
[9] Destino que aparece en el campo Dirección destino de la cabecera básica IPv6.

Cada cabecera de Extensión pueden aparecer solamente una vez, a excepción de la cabecera de opciones de Destino que puede aparecer dos veces, o antes de la cabecera de Enrutamiento o antes de la cabecera de nivel Superior.

Algunas cabeceras de Extensión se detallan a continuación:

2.5.4.1 **Cabecera de opciones salto a salto**: Esta cabecera de Extensión se identifica por el valor 0 en el campo Siguiente Cabecera de la cabecera básica IPv6[10], y se utiliza para transportar opciones especiales (alertas de enrutador y Jumbogramas) que se procesan en cada uno de los enrutadores a lo largo de la ruta de entrega de un paquete IPv6.

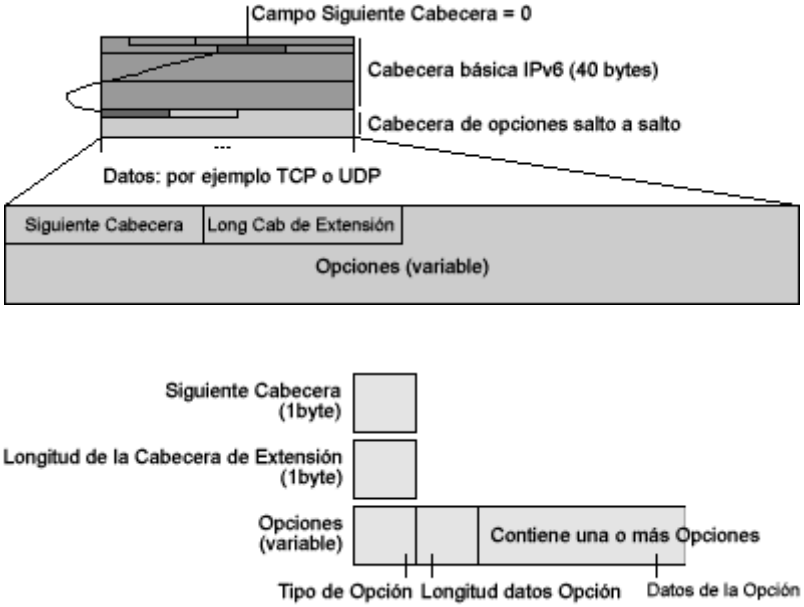


Figura 5 - Formato de la Cabecera de opciones salto a salto

La siguiente lista describe cada campo:

- Siguiente Cabecera: Este campo identifica el tipo de cabecera que sigue inmediatamente a la cabecera de opciones salto a salto.

- Longitud de la Cabecera de Extensión: Este campo identifica la longitud en unidades de 8bytes de la cabecera de opciones salto a salto.

- Opciones: La cabecera de opciones salto a salto y la cabecera de opciones de Destino, contienen una o más opciones codificadas con el formato: *tipo - longitud - valor*

[10] Si un paquete IPv6 contiene una Cabecera de opciones salto a salto, esta cabecera de Extensión está restringida a aparecer sólo inmediatamente después de una cabecera básica IPv6.

- *Tipo de Opción (1byte)*: Indicador del tipo de Opción.
- *Longitud Datos Opción (1byte)*: Es la longitud del campo de datos.
- *Datos de la Opción (variable)*: Son los datos. Contiene una o más Opciones.

Los valores del campo Tipo de Opción están codificados de tal manera que si un nodo IPv6 no reconoce uno de estos tipos, los 2 primeros bits le indican que debe hacer con el paquete:

00: Saltar el procesamiento de la cabecera de Extensión
01: Descartar el paquete IPv6
10: Descartar el paquete IPv6 y enviar un mensaje de error, independientemente de la dirección destino.
11: Descartar el paquete IPv6 y enviar un mensaje de error, si la dirección destino no es dirección de Multicast.

2.5.4.2 **Cabecera de Enrutamiento**: Esta cabecera de Extensión se identifica por el valor 43 en el campo Siguiente Cabecera de la cabecera anterior, y es utilizada por un enrutador origen para listar uno o más nodos intermedios que serán visitados en el camino hacía el destino de un paquete IPv6.

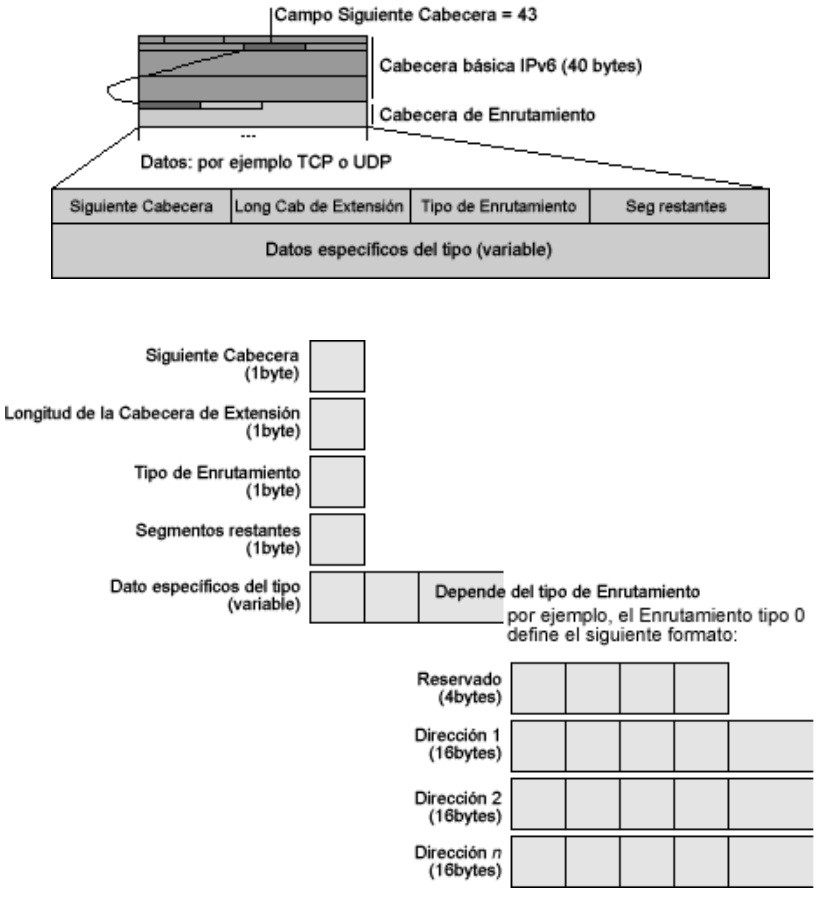


Figura 6 - Formato de la Cabecera de Enrutamiento

La siguiente lista describe cada campo:

- Siguiente Cabecera: Este campo identifica el tipo de cabecera que sigue inmediatamente a la Cabecera de Enrutamiento.

- Longitud de la Cabecera de Extensión: Este campo identifica la longitud en unidades de 8bytes de la Cabecera de Enrutamiento.

Tipo de Enrutamiento: Este campo identifica el tipo de Enrutamiento a utilizar. . La definición del Enrutamiento tipo 0 se explica en el RFC 2460, “Especificaciones del Protocolo Internet Versión 6 (IPv6)”.
Si al procesar esta cabecera el nodo encuentra que el “Tipo de enrutamiento” es un valor no reconocido se actuara de la siguiente forma:

- Si el número de segmentos restantes es 0 se ignorara la cabecera y se continuará con la siguiente.
- Si es diferente de 0, se descartará el paquete y se enviara un ICMP al origen indicando que hay problemas con el parámetro tipo de enrutamiento no reconocido.

- Segmentos restantes: Número de nodos intermedios explícitamente listados aún por visitar antes de alcanzar el destino final.

- Datos específicos del tipo: Depende del tipo de Enrutamiento.

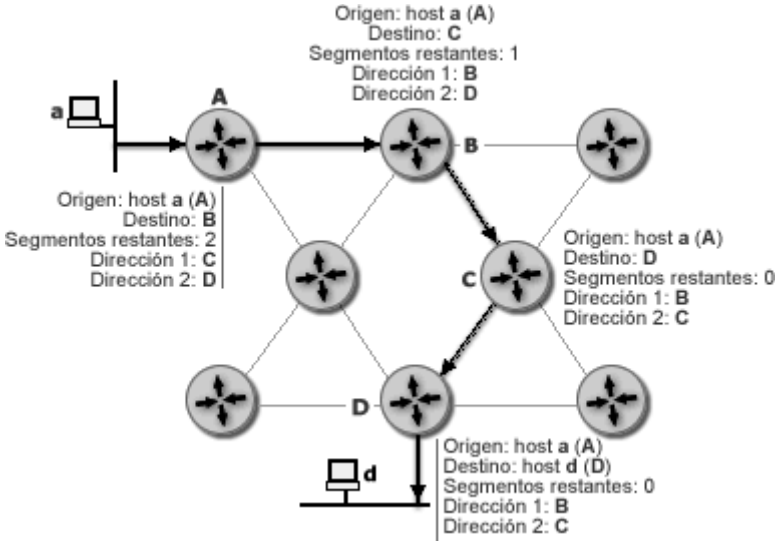


Figura 7 - Topología de enrutamiento IPv6 que utiliza Enrutamiento tipo 0

En la Figura 7, el enrutador origen **A** (host a) envía un paquete IPv6 al enrutador destino **D** (host d), utilizando una Cabecera de Enrutamiento que permite enrutar el paquete a través de nodos IPv6 intermedios que sirven de próximos destinos, los enrutadores **B** y **C**

2.5.4.3 **Cabecera de Fragmentación**: Esta cabecera de Extensión se identifica por el valor 44 en el campo Siguiente Cabecera de la cabecera anterior. Un nodo origen IPv6 que desea enviar un paquete a un nodo destino IPv6, utiliza el descubrimiento de MTU para determinar el tamaño máximo del paquete a enviar, pero si el tamaño del paquete es mayor al soportado por la MTU, el nodo origen fragmenta el paquete original en paquetes constitutivos más pequeños, identificando los paquetes mediante la cabecera de Fragmentación anexada a la cabecera básica IPv6 de cada paquete. En IPv6, un paquete no es fragmentado por nodos intermedios a lo largo de la ruta al destino. La fragmentación de paquetes se da únicamente en el nodo origen IPv6, y el reensamblado únicamente en el nodo destino IPv6.

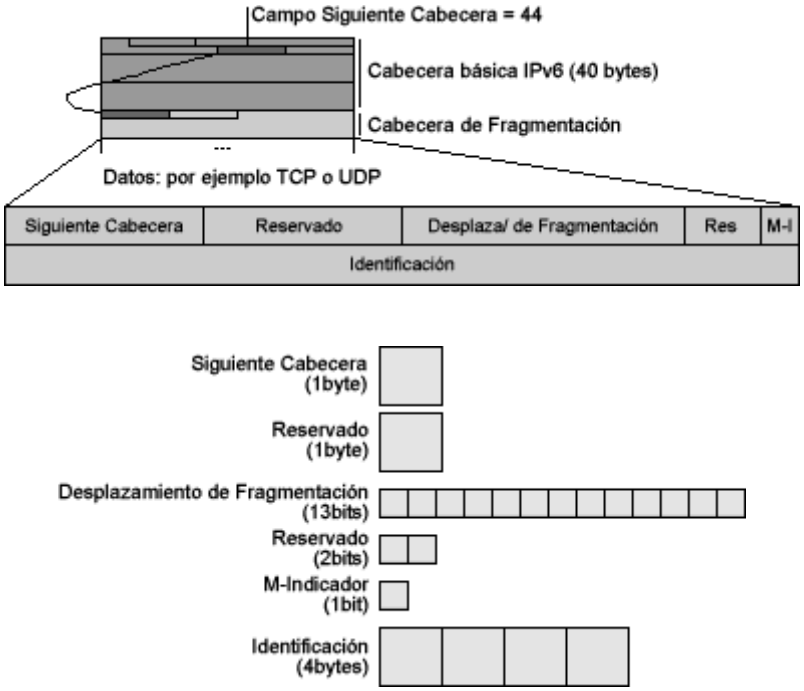


Figura 8 - Formato de la Cabecera de Fragmentación

La siguiente lista describe cada campo:

- Siguiente Cabecera: Este campo identifica el tipo de cabecera que sigue inmediatamente a la cabecera de Fragmentación.

- Reservado: No utilizado, este campo se fija a 0.

- Desplazamiento de Fragmentación “Fragment Offset”: Este campo indica la posición relativa de los datos de un fragmento en el paquete original.

- Reservado: No utilizado, este campo se fija a 0.

- M-Indicador: Un valor 1 en este campo indica más fragmentos, un valor 0 indica el último fragmento.

- Identificación: Número único, asignado por un nodo origen IPv6 que permite reensamblar el paquete IPv6 en el destino, de tal manera que los fragmentos de un mismo paquete deben tener el mismo valor de identificación.

2.6 Arquitectura de Direccionamiento en IPv6

A comienzos de la década de los años 80, cuando la arquitectura de direccionamiento IPv4 fue diseñada, era difícil imaginar que el espacio de direcciones podía algún día agotarse. Sin embargo, el crecimiento experimentado por Internet demostró que ya en los años 90 era necesario iniciar un conjunto de acciones orientadas a obtener un nuevo protocolo IP.

En consecuencia, la arquitectura de direccionamiento del protocolo IP de *próxima generación*[11] se diseño proporcionando compatibilidad e interoperabilidad con la arquitectura de red IPv4 existente, y facilitando además la coexistencia entre redes IPv4 e IPv6.

2.6.1 Espacio de Direccionamiento en IPv6

Una dirección IPv6 tiene una longitud de 128bits, 4 veces mayor que la longitud de una dirección IPv4. Una dirección de 32bits provee un espacio de direccionamiento de 2^{32} o 4,294,967,296 posibles direcciones IPv4. Una dirección de 128bits provee un espacio de direccionamiento de 2^{128} o 340,282,366,920,938,463,463,374,607,431,768,211456 ~ 3.4×10^{38} posibles direcciones IPv6. Así, la gran longitud de una dirección IPv6 esta diseñada para subdividirse en dominios de direccionamiento jerárquico que reflejan la topología actual de Internet y sus características de flexibilidad y escalabilidad.

2.6.2 Sintaxis de una dirección IPv6

Una dirección IPv6 es un identificador de 128bits para una interfaz o un conjunto de interfaces, que se representa como una serie de campos hexadecimales de 16bits separados por el símbolo (**:**). Ejemplo:

Inicialmente se tiene una dirección IPv6 en su forma binaria:

0010000111011010000000001101001100000000000000000010111100111011
0000001010101010000000001111111111111110001010001001110001011010

[11] RFC 2373 - RFC 3513 , “Arquitectura de Direccionamiento en IPv6”

Luego, la dirección de 128bits se divide en 8 campos de 16bits cada uno:

0010000111011010 0000000011010011 0000000000000000 0010111100111011
0000001010101010 0000000011111111 1111111000101000 1001110001011010

Finalmente cada campo de 16bits se convierte a una forma hexadecimal y se delimita utilizando el símbolo (**:**). El resultado es:

21DA:00D3:0000:2F3B:02AA:00FF:FE28:9C5A >>
21DA:D3:0:2F3B:2AA:FF:FE28:9C5A

Compresión de Ceros

IPv6 utiliza convenciones que permiten representar una dirección de una manera más simple:

Los ceros de mayor peso en cada campo hexadecimal son opcionales y se pueden eliminar.

Ejemplo 1: 2031:**0000**:130F:**0000**:**0000**:**0**9C0:876A:130B >>
2031:0:130F:0:0:9C0:876A:130B
Ejemplo 2: FF02:30:**000**0:5 >> FF02:30:0: 5

El símbolo (**::**) representa sucesivos campos hexadecimales de valor cero. Sin embargo, este símbolo únicamente puede aparecer una vez en la dirección IPv6.

Ejemplo 1: 2031:0:130F:**0**:**0**:9C0:876A:130B >>
2031:0:130F::9C0:876A:130B

Ejemplo 2: FF01:**0**:**0**:**0**:**0**:**0**:**0**:1 >> FF01::1

Para determinar la cantidad de bits “0” representados por el símbolo (**::**), es necesario precisar la cantidad de campos hexadecimales de la dirección IPv6 comprimida, restar este valor de 8 y multiplicar el resultado por 16.

Ejemplo 1: La dirección 2031:0:130F::9C0:876A:130B posee 6 campos hexadecimales. Entonces [8 - 6]16 = 32bits consecutivos de valor “0” se representan por el símbolo (**::**)

Ejemplo 2: La dirección FF01::1 posee 2 campos hexadecimales.

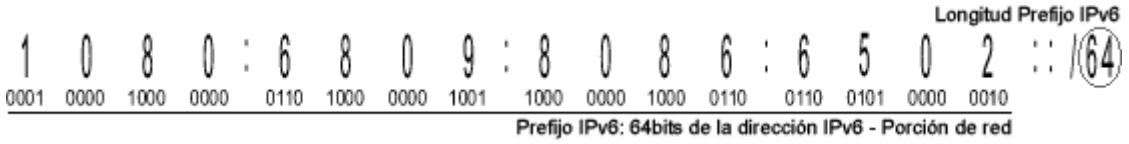
Entonces [8 - 2]16 = 96bits consecutivos de valor “0” se representa por el símbolo (**::**).

2.6.3 **Prefijos de direcciones IPv6**

Un prefijo es una parte de la dirección IPv6 que indica los bits que representan el *Format Prefix*[12] o los bits que representan el Identificador de red. La longitud de un prefijo es un valor decimal que indica cuántos bits de la parte izquierda[13] de una dirección IPv6 componen el prefijo.

Estos prefijos utilizan la notación: *dirección-IPv6* **/** *longitud-del-prefijo*, similar a la notación *Classless Inter-Domain Routing*, Enrutamiento entre dominios sin clase (CIDR) en IPv4.

Ejemplo: La notación 1080:6809:8086:6502::/64 define un prefijo IPv6 válido, con una longitud de prefijo de 64bits.



Por lo tanto, la notación de una dirección IPv6 completa indicando la porción de red, es de la forma: **1080:6809:8086:6502**:1234:1234:1234:1234/64

2.6.4 **Tipos de direcciones IPv6**

Las direcciones IPv6 se clasifican en 3 tipos:

- Unicast: Identificador para una única interfaz. Un paquete IPv6 enviado a una dirección Unicast es entregado sólo a la interfaz identificada con dicha dirección.

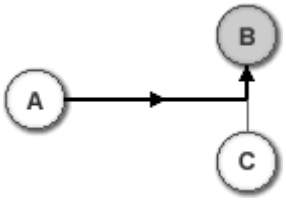


Figura 9 - Comportamiento Unicast

- Anycast: Identificador para múltiples interfaces. Un paquete IPv6 enviado a una dirección Anycast es entregado a una única interfaz, la interfaz más cercana identificada con dicha

[12] Este concepto se explica más adelante
[13] Los bits más significativos de la dirección IPv6

dirección. Esta interfaz es definida en términos de la medida de la distancia de los protocolos de enrutamiento.

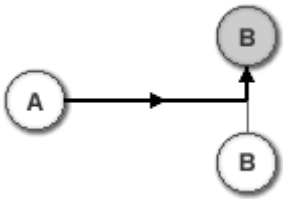


Figura 10 - Comportamiento Anycast

- Multicast: Identificador para múltiples interfaces. Un paquete IPv6 enviado a una dirección Multicast es entregado a todas las interfaces identificadas por dicha dirección. En IPv6, la función de las direcciones de *broadcast* es sustituida por direcciones Multicast.

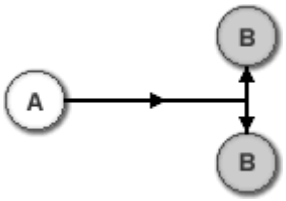


Figura 11 - Comportamiento Multicast

El tipo específico al que pertenece una dirección IPv6 esta dado por los primeros bits en el prefijo de la dirección. Estos bits forman el denominado *Format Prefix (FP)*, que es de longitud variable, según el tipo de dirección que se trate.

La asignación de FP's se muestra en la Tabla 2:

Asignación	**Format Prefix**
Reservado	0000 0000
Sin asignar	0000 0001
Reservado para asignación NSAP[14]	0000 001
Reservado para asignación IPX[15]	0000 010
Sin asignar	0000 011
Sin asignar	0000 1
Sin asignar	0001
Direcciones Unicast Globales Agregables	001
Sin asignar	010
Sin asignar	011
Sin asignar	100
Sin asignar	101
Sin asignar	110
Sin asignar	1110
Sin asignar	1111 0
Sin asignar	1111 10
Sin asignar	1111 110
Sin asignar	1111 1110 0

[14] NSAP - Network Service Access Point
[15] IPX - Internetwork Packet Exchange Protocol

Direcciones Unicast Locales de Enlace	1111 1110 10
Direcciones Unicast Locales de Sitio	1111 1110 11
Direcciones Multicast	1111 1111

Tabla 2 **-** Asignación de direcciones IPv6

Nota: Se puede distinguir una dirección Unicast de una Multicast por el valor de sus 8bits más significativos, 11111111 o FF en hexadecimal, identifican una dirección Multicast.

De acuerdo con la Tabla 2, en la actualidad está asignado aproximadamente el 15% del espacio de direccionamiento IPv6, distribuido entre direcciones de agregación, de uso local y Multicast, reservándose el 85% restante para uso futuro.

2.6.5 **Características de las direcciones IPv6**

- Las direcciones IPv6 identifican interfaces no hosts ni enrutadores.
- Un nodo IPv6 se identifica por una dirección Unicast asignada a una de sus interfaces.
- Una interfaz puede tener asociadas múltiples direcciones IPv6 de cualquier tipo.
- Todas las interfaces tienen por lo menos una dirección IPv6 Local de Enlace (Link-Local).

Existen 1 excepción a los anteriores criterios:

- Una misma dirección o un conjunto de direcciones Unicast, se pueden asignar a múltiples interfaces siempre que la implementación trate a estas interfaces como una única interfaz cuando las presenta a Internet.

2.7 **Direcciones Unicast**

Las direcciones Unicast pueden estructurarse de diferentes maneras:

- Dirección Global Agregable
- Dirección Local de Sitio (Site-Local)
- Dirección Local de Enlace (Link-Local)
- Direcciones Especiales

En consecuencia, éstas direcciones IPv6 tienen ámbitos de acción[16]: *Local de Enlace - Local de Sitio - Global*

[16] Ámbito de acción, área en la cual una dirección puede utilizarse como identificador único de una o varias interfaces.

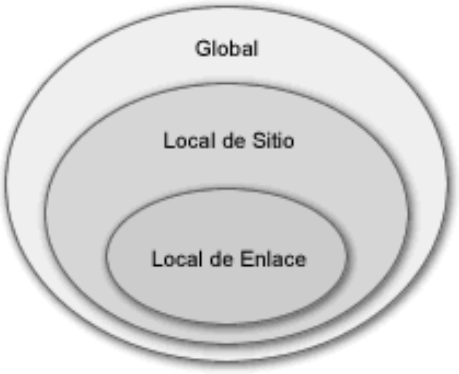


Figura 12 - Ámbitos de acción de una dirección IPv6

2.7.1 **Dirección Global Agregable**[17]

Este tipo de dirección IPv6 es equivalente a una dirección pública IPv4, ya que permite un direccionamiento global, alcanzando cualquier destino IPv6 en Internet.

A diferencia de la arquitectura IPv4 de Internet, la cual provee una organización pseudo jerárquica[18] del direccionamiento y enrutamiento, la arquitectura IPv6 de Internet utiliza el concepto de agregación[19] de prefijos para facilitar una organización jerárquica con múltiples niveles, limitando así el número de entradas de rutas en las tablas de enrutamiento de Internet.

El modelo de direccionamiento jerárquico en IPv6 define 3 niveles:

- Topología Pública[20]: Conjunto de *Proveedores de Acceso* y *Puntos de Intercambio* que ofrecen servicios de tránsito en Internet IPv6.

- Topología de Sitio: Topología local a una organización que no ofrece servicios de tránsito a nodos exteriores a la organización.

- Identificador de Interfaz: Identificador único asignado a cada interfaz en el ámbito de un enlace.

[17] RFC 2374 - RFC 3587, “Formato de Direcciones Unicast Globales Agregables”
[18] IPv4 maneja una jerarquía de 2 niveles: Identificador de red, Identificador de host
[19] IPv6 define 2 tipos de agregación: por Proveedor de Acceso y por Punto de Intercambio (Exchange)
[20] ISP’s IPv6: Proveedores de Servicios en Internet IPv6

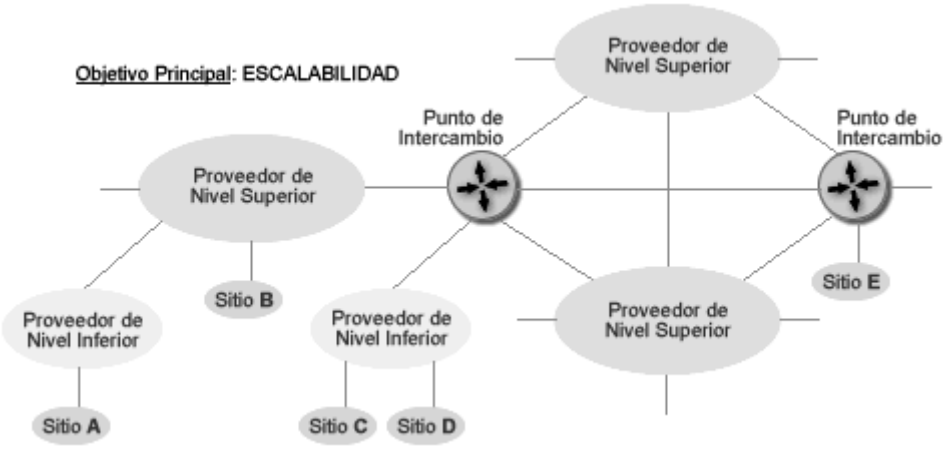


Figura 13 - Enrutamiento jerárquico en IPv6

En la Figura 13, se muestra una topología global de direccionamiento por niveles en IPv6, utilizando direcciones Globales Agregables, las cuales soportan Proveedores de Nivel Superior, Puntos de Intercambio, Proveedores de Nivel Inferior y Sitios Finales.

La máxima autoridad competente que plantea las reglas para la administración del modelo de direccionamiento jerárquico en IPv6, es la *Internet Assigned Numbers Authority*, Agencia de asignación de números Internet (IANA), y en consecuencia, ésta define un Sistema jerárquico de Registro en Internet, que para IPv6 consiste de los siguientes niveles:

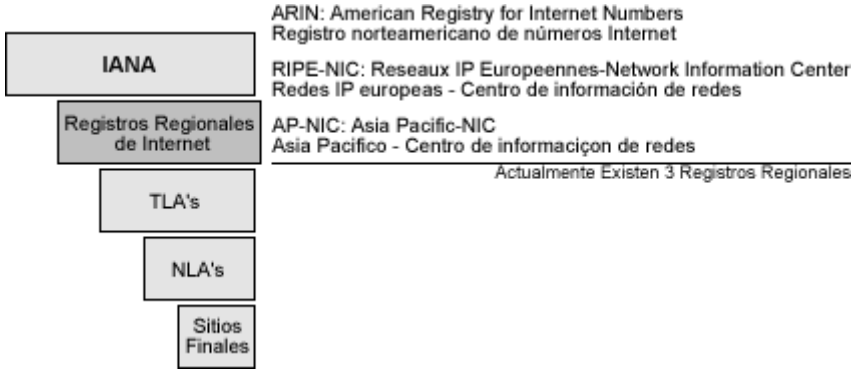


Figura 14 - Sistema jerárquico de Registro en Internet IPv6

Los niveles TLA's y NLA's[21] en la Figura 14, se comprenden con facilidad, al analizar el formato de una dirección Global Agregable, el cual se ilustra a continuación:

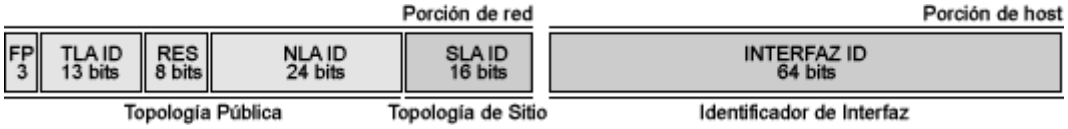


Figura 15 - Formato de una Dirección Global Agregable

[21] RFC 2450, "Normas de asignación de TLA's y NLA's"

La siguiente lista describe cada campo:

- FP, *Format Prefix*: 2000::/3 (001)

- TLA ID - *Top Level Aggregation Identifier*, Identificador de Agregación de Nivel Superior: El Identificador de TLA se establece bajo la autoridad de un Registro Regional de Internet, y representa el nivel superior del modelo de direccionamiento jerárquico en IPv6. Un enrutador IPv6 situado en este nivel tiene en su tabla de enrutamiento una entrada por cada TLA ID activo, y probablemente entradas adicionales dependiendo de la topología de red. La longitud del campo TLA ID permite 2^{13} o 8,192 identificadores de TLA.

- RES: Reservado para permitir un adecuado crecimiento de los campos TLA ID y NLA ID.

- NLA ID - *Next Level Aggregation Identifier*, Identificador de Agregación de Nivel Próximo: El Identificador de NLA se establece bajo la autoridad de una organización que administra un TLA ID, y se utiliza para crear una estructura de direccionamiento e identificar sitios. Una organización que administre un Identificador de NLA, puede asignar la parte superior del NLA ID para crear una jerarquía apropiada en su red, y además puede utilizar los bits restantes del campo NLA ID para identificar sitios a quienes desee dar servicio.

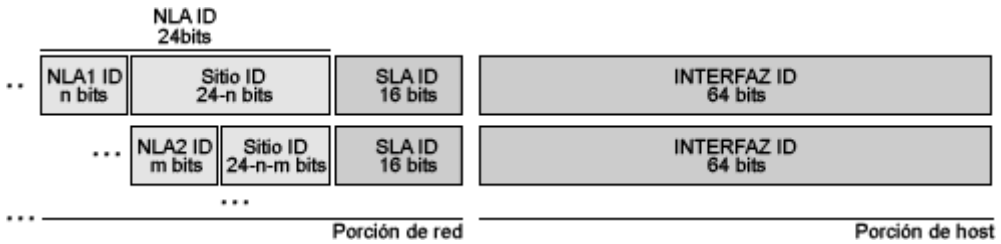


Figura 16 - Jerarquía al interior de un campo NLA ID

- SLA ID - *Site Level Aggregation Identifier*, Identificador de Agregación de Nivel de Sitio: El Identificador de SLA es utilizado por una organización final para crear su propia estructura de direccionamiento e identificar sus subredes. Así, una organización puede utilizar los 16bits del campo SLA ID para crear 2^{16} o 65,536 subredes, o al igual que con el Identificador de NLA, puede crear al interior de éste campo múltiples niveles de direccionamiento y una infraestructura de enrutamiento eficiente.

- INT ID - Identificador de Interfaz: Identificador único asignado a cada interfaz en el ámbito de un enlace.

2.7.2 **Dirección Local de Sitio (Site-Local)**

Este tipo de direcciones Unicast son equivalentes a las direcciones privadas 10.0.0.0/8, 172.16.0.0/12, y 192.168.0.0/16 en redes IPv4, y están diseñadas para ser utilizadas localmente por una organización que espera en un futuro migrar a un esquema de direccionamiento global.

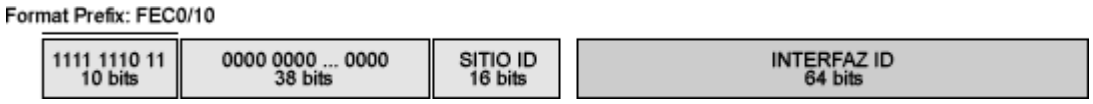


Figura 17 - Formato de una Dirección Local de Sitio

Cuando una organización está lista para migrar a una conexión global, los bits que identifican el *Format Prefix (FP)* de la dirección Local de Sitio así como el campo inicializado a ceros, se sustituyen por el prefijo de la topología pública a utilizar. Entonces los 16bits asignados a la creación de subredes en la organización, son ahora los 16bits del campo SLA ID de una dirección Unicast Global Agregable.

Nota: Un enrutador IPv6 nunca reenvía trafico local de sitio a un destino fuera del ámbito del sitio.

2.7.3 **Dirección Local de Enlace (Link-Local)**

Este tipo de dirección Unicast se asigna automáticamente a una interfaz, asociando el *Format Prefix (FP)* de la dirección Local de Enlace y el Identificador de Interfaz en su formato EUI-64. Un nodo IPv6 utiliza una dirección Local de Enlace cuando desea comunicarse con un nodo vecino en el mismo enlace para propósitos de Descubrimiento de vecinos y/o Autoconfiguración de direcciones.

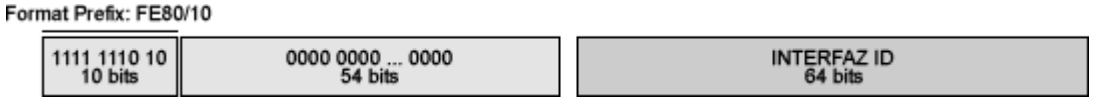


Figura 18 - Formato de una Dirección Local de Enlace

Nota: Un enrutador IPv6 nunca reenvía tráfico local de enlace a un destino fuera del ámbito del enlace.

2.7.4 **Direcciones Especiales**

2.7.4.1 **Dirección sin especificar (Unspecified)**: Es la dirección Unicast 0:0:0:0:0:0:0:0, también representada de la forma 0::0 o más comúnmente ::/128. Esta dirección indica ausencia de dirección y no puede asignarse a una interfaz en un nodo IPv6.

Nota: La dirección ::/128 nunca debe utilizarse como dirección destino en paquetes IPv6 o en la Cabecera de Enrutamiento IPv6.

2.7.4.2 **Dirección de Loopback**[22]: Es la dirección Unicast 0:0:0:0:0:0:0:1 o simplemente ::1. Esta dirección identifica una interfaz local virtual, el interfaz de Loopback.

Nota: La dirección ::1 nunca debe utilizarse como dirección Origen o Destino en paquetes IPv6.

2.8 Direcciones Anycast

Este tipo de dirección IPv6 le permite a un origen conectarse con cualquier interfaz perteneciente a un grupo de interfaces (normalmente en nodos IPv6 diferentes), a través de una única dirección. Una dirección Anycast puede ser utilizada como parte de una secuencia de enrutamiento al emplearse como dirección intermedia en una cabecera de Enrutamiento, así un nodo IPv6 puede seleccionar cual Proveedor de Servicios de Internet transportará su tráfico.

No existe diferencia sintáctica entre las direcciones Anycast con las direcciones Unicast, ya que las direcciones Anycast se asignan desde el espacio de direccionamiento Unicast. El ámbito de las direcciones Anycast es igual al UniCast, así, pueden existir direcciones Anycast de ámbitos Local de Enlace, Local de Sitio y Global.

2.9 Direcciones Multicast

Este tipo de dirección IPv6 es un identificador único para un grupo de interfaces[23] (normalmente en nodos IPv6 diferentes). Las direcciones Multicast tienen el siguiente formato:

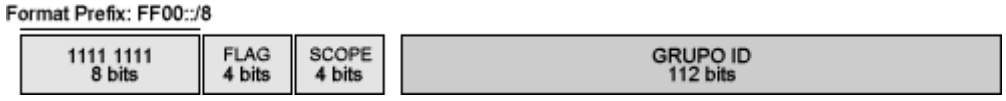


Figura 19 - Formato de una Dirección Multicast

Los bits 11111111 o FF en hexadecimal al inicio de la dirección, identifican una dirección del tipo Multicast.

La siguiente lista describe cada campo:

- FLAG: Es un conjunto de 4 indicadores, 000T, donde los 3 primeros bits están reservados y se inicializan a cero, y el bit T anuncia direcciones Multicast permanentes (0) asignadas por la IANA, o direcciones Multicast temporales (1).

[22] Es similar a la dirección de Loopback 127.0.0.1 en IPv4
[23] Una interfaz puede pertenecer a cualquier número de grupos Multicast

- SCOPE: Los 4bits del campo SCOPE limitan el ámbito del grupo Multicast. Los valores predominantes de éste campo son 1 (ámbito Local de Nodo), 2 (ámbito Local de Enlace), 5 (ámbito Local de Sitio), 8 (ámbito Local de Organización) y E (ámbito global).

- GRUPO ID: El campo Identificador de Grupo, distingue grupos Multicast y es único en un determinado ámbito.

Las direcciones Multicast desde FF00:: a FF0F::, son reservadas y se asignan a funciones específicas. Para identificar nodos en los ámbitos Local de Nodo y Local de Enlace, las siguientes direcciones Multicast son definidas:

FF01::1 (dirección Multicast todos los nodos - ámbito Local de Nodo)
FF02::1 (dirección Multicast todos los nodos - ámbito Local de Enlace)

Para identificar enrutadores en los ámbitos Local de Nodo, Local de Enlace y Local de Sitio, las siguientes direcciones Multicast son definidas:

FF01::2 (dirección Multicast todos los enrutadores - ámbito Local de Nodo)
FF02::2 (dirección Multicast todos los enrutadores - ámbito Local de Enlace)
FF05::2 (dirección Multicast todos los enrutadores - ámbito Local de Sitio)

Además, IPv6 define la dirección Multicast Nodo Solicitado FF02::1:FFXX:XXXX, para utilizarse con el protocolo de Descubrimiento de vecinos, en lugar de emplear direcciones Multicast FF02::1.

2.10 Direcciones IPv6 requeridas por un Nodo

2.10.1 Direcciones IPv6 requeridas por un Host

- Dirección Local de Enlace por cada interfaz
- Dirección Local de Sitio, dirección Global Agregable por cada interfaz *(opcional)*
- Dirección de Loopback
- Dirección Multicast todos los nodos - ámbito Local de Nodo
- Dirección Multicast todos los nodos - ámbito Local de Enlace
- Dirección Multicast Nodo Solicitado por cada dirección Unicast o Anycast asignada
- Dirección Multicast de los grupos a los que pertenece

2.10.2 Direcciones IPv6 requeridas por un Enrutador

- Dirección Local de Enlace por cada interfaz

- Dirección Local de Sitio, dirección Global Agregable por cada interfaz *(opcional)*
- Dirección de Loopback
- Dirección Anycast del enrutador de la subred[24]
- Dirección Multicast todos los nodos - ámbito Local de Nodo
- Dirección Multicast todos los nodos - ámbito Local de Enlace
- Dirección Multicast todos los enrutadores - ámbito Local de Nodo
- Dirección Multicast todos los enrutadores - ámbito Local de Enlace
- Dirección Multicast todos los enrutadores - ámbito Local de Sitio
- Dirección Multicast Nodo Solicitado por cada dirección Unicast o Anycast asignada
- Dirección Multicast de los grupos a los que pertenece

2.11 Protocolo ICPMv6

El Protocolo de mensajes de control en Internet (ICMP) en IPv6 funciona de manera similar a ICMP en IPv4. ICMPv6 genera mensajes de error, tales como mensajes ICMPv6 de destino inalcanzable, mensajes de información como ICMPv6 echo request y mensajes de réplica, que tienen valores de tipo de mensaje entre 128 y 255.

Adicionalmente, los paquetes ICMPv6 son utilizados en los procesos de descubrimiento de vecinos, descubrimiento de ruta MTU y por el protocolo MLD (multicast listener discovery).

En la siguiente tabla se observan los diferentes tipos de mensajes ICMPv6 con su respectivo valor:

Descripción	**Tipo**
Destination Unreacheable, Destino inalcanzable	1
Packet too big, Paquete demasiado grande	2
Time exceeded, Tiempo excedido	3
Parameter problem, Problemas de parámetros	4
Echo request	128
Echo reply	129
Router Solicitation, Solicitud de enrutador	133
Router Advertisement, Anuncio de enrutador	134
Neighbor Solicitation, Solicitud de vecino	135
Neighbor Advertisement, Anuncio de vecino	136
Redirect, Redirección	137
ICMP Node Information Query, Solicitud de información de nodo ICMP	139
ICMP Node Information Response, Respuesta de información de nodo ICMP	140
Inverse Neighbor Discovery Solicitation Message, Mensaje de solicitud de descubrimiento de vecino inverso	141
Mobile Prefix Solicitation, Solicitud de prefijo móvil	145
Mobile Prefix Advertisement, Anuncio de prefijo móvil	146

Tabla 3 - Mensajes ICMPv6

[24] Dirección Anycast requerida para cada subred. Su sintaxis es equivalente al prefijo que especifica el enlace correspondiente de la dirección Unicast, siendo el Identificador de Interfaz igual a cero.

Un valor de 58 en el campo de Siguiente Cabecera de la Cabecera básica de un paquete IPv6 identifica un paquete ICMPv6. Los paquetes ICMPv6 son como un paquete de la capa de transporte, en el sentido que el paquete ICMPv6 sigue todas las Cabeceras de Extensión y es la última pieza de información en el paquete IPv6.

Dentro del paquete ICMPv6, los campos *tipo* y *código* identifican paquetes ICMPv6 específicos. El valor en el campo *checksum* es derivado de los campos en el paquete ICMPv6 y la Cabecera de IPv6. El campo de datos ICMP contiene información de error y diagnóstico relevante para el procesamiento del paquete IPv6.

Similar a ICMPv4, ICMPv6 es muchas veces bloqueado por políticas de seguridad implementadas en firewalls debido a los ataques basados en ICMP. Sin embargo, ICMPv6 tiene la capacidad de utilizar cifrado y autenticación IPsec. Estos servicios de seguridad disminuyen las posibilidades de un ataque basado en ICMPv6.

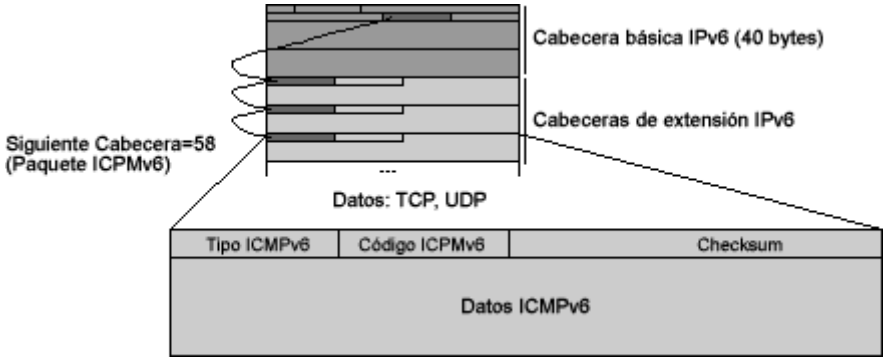


Figura 20 - Formato de un paquete ICMPv6

2.12 Protocolo de Descubrimiento de vecinos

En IPv6, el protocolo equivalente, en cierto modo, a ARP en IPv4, es el que denominamos “Descubrimiento de vecinos”. Sin embargo, incorpora también la funcionalidad de otros protocolos IPv4, como “ICMP Router Discovery” y “ICMP Redirect”. Tal como se indica, consiste en el mecanismo por el cual un nodo que se incorpora a una red, descubre la presencia de otros, en su mismo enlace, para determinar sus direcciones en la capa de enlace para localizar los enrutadores, y para mantener la información de conectividad (“reachability”) acerca de las rutas a los vecinos activos.

El protocolo ND, también se emplea para mantener limpios los “caches” donde se almacena la información relativa al contexto de la red a la que esta conectado un nodo IPv6, y por tanto para detectar cualquier cambio en la misma. Cuando un enrutador, o una ruta hacia él, falla, un host buscará alternativas funcionales. ND emplea los mensajes de ICMPv6, incluso a través de mecanismos de Multicast en la capa de enlace, para algunos de sus servicios.

El protocolo ND es bastante completo y sofisticado, ya que es la base para permitir el mecanismo de autoconfiguración en IPv6. Define, entre otros, mecanismos para descubrir enrutadores, prefijos y parámetros, autoconfiguración de direcciones, resolución de direcciones, determinación del siguiente salto, detección de nodos no alcanzables, detección de direcciones duplicadas o cambios, redirección, balanceo de carga entrante, direcciones Anycast y anuncio de proxies.

ND define cinco tipos de paquetes ICMPv6:

- Solicitud de Enrutador (Router Solicitation): generado por una interfaz cuando es activada, para solicitar a los enrutadores que se "anuncien" inmediatamente.
  Tipo en paquete ICMPv6 = 133.

- Anuncio de Enrutador (Router Advertisement): generado por los enrutadores periódicamente (entre cada 4 y 1800 segundos) o como consecuencia de una "solicitud de enrutador", a través de Multicast, para informar de su presencia, así como de otros parámetros de enlace y de Internet, como prefijos, tiempos de vida, configuración de direcciones, límite de salto sugerido, etc. Es fundamental para permitir la renumeración.
  Tipo en paquete ICMPv6 = 134.

- Solicitud de Vecino (Neighbor Solicitation): generado por los nodos para determinar la dirección en la capa de enlace de sus vecinos, o para verificar que el nodo vecino sigue activo (es alcanzable), así como para detectar las direcciones duplicadas.
  Tipo en paquete ICMPv6 = 135.

- Anuncio de Vecino (Neighbor Advertisement): generado por los nodos como respuesta a la "solicitud de vecino", o bien para indicar cambios de direcciones en la capa de enlace.
  Tipo en paquete ICMPv6 = 136.

- Redirección (Redirect): generado por los enrutadores para informar a los hosts de un salto mejor para llegar a un determinado destino. Equivalente, en parte a "ICMP redirect".
  Tipo en paquete ICMPv6 = 137.

El protocolo ND, frente a los mecanismos existentes en IPv4, reporta numerosas ventajas:

- El descubrimiento de enrutadores es parte de la base del protocolo, no es preciso recurrir a los protocolos de enrutamiento.

- El anuncio de enrutador incluye las direcciones de la capa de enlace, no es necesario ningún intercambio adicional de paquetes para su resolución, incluye los prefijos para el

enlace, por lo que no hay necesidad de un mecanismo adicional para configurar la máscara de red. Además permite la autoconfiguración de direcciones.

- Los enrutadores pueden anunciar a los nodos del mismo enlace el MTU.

- Se pueden asignar múltiples prefijos al mismo enlace y por defecto los nodos aprenden todos los prefijos por el anuncio de enrutador. Sin embargo, los enrutadores pueden ser configurados para omitir parte o todos los prefijos en el anuncio, de forma que los nodos consideren que los destinos están fuera del enlace; de esta forma, enviarán el tráfico a los enrutadores, quienes a su vez lo redireccionarán según corresponda.

- A diferencia de IPv4, en IPv6 el receptor de una redirección asume que el siguiente salto está en el mismo enlace. Se prevé una gran utilidad en el sentido de no ser deseable o posible que los nodos conozcan todos los prefijos de los destinos en el mismo enlace (enlaces sin multidifusión y media compartida).

- La detección de vecinos no alcanzables es parte de la base de mejoras para la robustez en la entrega de paquetes frente a fallos en enrutadores, particiones de enlaces, nodos que cambian sus direcciones, nodos móviles, etc.

- En ND se puede detectar fallos de la mitad del enlace, es decir, con conectividad en un sólo sentido, evitando el tráfico hacia ellos.

- A diferencia de IPv4, no son precisos campos de preferencia (para definir la “estabilidad” de los enrutadores). La detección de vecinos no alcanzables sustituirá los caminos desde enrutadores con fallos a otros activos.

- El uso de direcciones de enlace local para identificar enrutadores permite a los nodos que mantengan su asociación con los mismos, en el caso de que se realice una renumeración para usar nuevos prefijos globales.

- El límite de saltos es siempre igual a 255, lo que evita que haya envíos accidentales o intencionados desde nodos fuera del enlace, dado que los enrutadores decrementan automáticamente este campo en cada salto.

- Al realizar la resolución de direcciones en la capa ICMP, se independiza el protocolo del medio, permitiendo mecanismos de autenticación y seguridad normalizados.

2.13 **Autoconfiguración en IPv6**

La autoconfiguración es el conjunto de pasos por los cuales un host decide como autoconfigurar sus interfaces en IPv6. Este mecanismo es el que permite afirmar que IPv6 es "Plug and Play". El proceso incluye la creación de una dirección Local de Enlace, verificación de que no esta duplicada en dicho enlace y determinación de la información que ha de ser autoconfigurada (direcciones y otra información).

Las direcciones pueden obtenerse de forma totalmente manual, mediante DHCPv6 (con control de estado o configuración predeterminada), o de forma automática (sin control de estado o configuración automático sin intervención).

DHCPv6 define el proceso de generar una dirección Local de Enlace, direcciones Globales Agregables y Locales de Sitio, mediante el procedimiento automático. También define el mecanismo para detectar direcciones duplicadas.

La autoconfiguración sin control de estado (stateless), no requiere ninguna configuración manual del nodo, configuración mínima de enrutadores, y no precisa servidores adicionales. Permite a un nodo generar su propia dirección mediante una combinación de información disponible localmente e información anunciada por los enrutadores. Los enrutadores anuncian los prefijos que identifican la subred asociada con el enlace, mientras el host genera un "Identificador de Interfaz", que identifica de forma única la interfaz en la subred. La dirección se compone por la combinación de ambos campos. En ausencia de un enrutador, el host sólo puede generar la dirección de Local de Enlace, aunque esto es suficiente para permitir la comunicación entre nodos conectados al mismo enlace.

En la autoconfiguración con control de estado (statefull), el nodo obtiene la dirección de la interfaz y/o la información y parámetros de configuración desde un servidor. Los servidores mantienen una base de datos con las direcciones que han sido asignadas a cada nodo.

Ambos tipos de autoconfiguración se complementan. Un host puede usar autoconfiguración sin control de estado para generar su propia dirección y obtener el resto de parámetros mediante autoconfiguración con control de estado.

El mecanismo de autoconfiguración sin control de estado se emplea cuando no importa la dirección exacta que se asigna a un host, sino tan sólo asegurarse que es única y correctamente enrutable. El mecanismo de autoconfiguración predeterminada, por el contrario, asegura que cada host tiene una determinada dirección, asignada manualmente.

La autoconfiguración esta diseñada para hosts, no para enrutadores, aunque esto no implica que parte de la configuración de los enrutadores también pueda ser realizada automáticamente.

2.13.1 **Autoconfiguración sin Control de Estado (stateless)**

El procedimiento de autoconfiguración sin control de estado (stateless o descubrimiento automático), ha sido diseñado con las siguientes premisas:

- Evitar la configuración manual de dispositivos antes de su conexión a la red. Se requiere, en consecuencia, un mecanismo que permita a los host obtener o crear direcciones únicas para cada una de sus interfaces, asumiendo que cada interfaz puede proporcionar un identificador único para si misma (identificador de interfaz). En el caso más simple, el identificador de interfaz consiste en la dirección de la capa de enlace, de dicha interfaz. El identificador de interfaz puede ser combinado con un prefijo, para formar la dirección.

- Las pequeñas redes o sitios con máquinas conectadas a un único enlace no deberían requerir la presencia de un servidor “con control de estado” o enrutador, como requisito para comunicarse. Para obtener, en este caso, características “Plug & Play”, se emplean las direcciones de Locales de Enlace, dado que tienen un prefijo perfectamente conocido que identifica el único enlace compartido, al que se conectan todos los nodos. Cada dispositivo forma su dirección de Local de Enlace anteponiendo el prefijo de enlace local a su identificador de interfaz.

- En el caso de redes o sitios grandes, con múltiples subredes y enrutadores, tampoco se requiere la presencia de un servidor de configuración de direcciones “con control de estado”, ya que los host han de determinar, para generar sus direcciones Globales Agregables o Locales de Enlace, los prefijos que identifican las subredes a las que se conectan. Los enrutadores generan mensajes periódicos de anuncio, que incluyen opciones como listas de prefijos activos en los enlaces.

- La configuración de direcciones debe de facilitar la renumeración de los dispositivos de un sitio, por ejemplo, cuando se desea cambiar de Proveedor de Servicios. La renumeración se logra al permitir que una misma interfaz pueda tener varias direcciones.

- Sólo es posible utilizar este mecanismo en enlaces capaces de manejar funciones Multicast, y comienza, por tanto, cuando es iniciada o activada una interfaz que permite Multicast.

Los pasos básicos para la autoconfiguración, una vez la interfaz ha sido activada, son:

- Se genera la dirección “tentativa” Local de Enlace, como se ha descrito antes.

- Verificar que dicha dirección “tentativa” puede ser asignada (no esta duplicada en el mismo enlace).

- Si esta duplicada, la autoconfiguración se detiene, y se requiere un procedimiento manual (por ejemplo, usando otro identificador de interfaz).

- Si no esta duplicada, la conectividad a nivel IP se ha logrado, al asignarse definitivamente dicha dirección “tentativa” a la interfaz en cuestión.

- Si se trata de un host, se interroga a los posibles enrutadores para indicar al host que debe hacer.

- Si no hay enrutadores, se invoca el procedimiento de autoconfiguración “con control de estado”.

- Si hay enrutadores, estos contestarán indicando fundamentalmente, como obtener las direcciones si se ha de utilizar el mecanismo “con control de estado”, u otra información, como tiempos de vida, etc.

2.13.2 **Autoconfiguración con Control de Estado** (Statefull)

En el modelo de auto configuración con estado, los nodos obtienen direcciones de interfaz o información de configuración y parámetros de un servidor. Los servidores mantienen una base de datos que revisa cuales direcciones han sido asignadas. El protocolo de autoconfiguración con estado permite a los nodos obtener direcciones y otra información de configuración desde un servidor.

La configuración con estado es utilizada cuando un sitio requiere un control más preciso sobre direcciones exactas asignadas. Autoconfiguración con estado y sin estado puede ser utilizada simultáneamente. El administrador del sitio especifica cual tipo de auto configuración usar a través del envío de campos apropiados en los mensajes de advertencia del enrutador.

Cada dirección tiene un tiempo de vida asociado que indica el tiempo que la dirección está en una interfaz. Cuando el tiempo de vida termina, la conexión y la dirección se hacen inválidas y la dirección puede ser reasignada para otra interfaz en cualquier parte.

2.13.3 **Anuncios de Enrutador**

La próxima fase de la autoconfiguración consiste en obtener un aviso de enrutador o determinar que no hay enrutadores presentes. Si hay enrutadores presentes, estos mandan un anuncio de enrutador que especifica que tipo de autoconfiguración debería emplear un host. Si no hay enrutadores presentes, la autoconfiguración con estado es invocada. Los enrutadores envían un anuncio de enrutador periódicamente. Sin embargo, el retardo entre sucesivos avisos es generalmente mas largo que lo que un host que emplea autoconfiguración pueda esperar. Para obtener un aviso rápido, un host envía una o más solicitudes de enrutador para todos los enrutadores del grupo Multicast. Los avisos de enrutador contienen dos banderas que indican que tipo de autoconfiguración con estado debería ser aplicada. Una bandera de configuración de gestión de dirección indica como un host debe utilizar autoconfiguración con estado para obtener direcciones. La otra bandera de configuración con estado indica como un host debe utilizar autoconfiguración con estado para obtener información adicional, excepto direcciones.

2.14 **Mecanismos de Transición**

La clave para la transición es la compatibilidad con la base instalada de dispositivos IPv4. Esta afirmación define un conjunto de mecanismos que los hosts y enrutadores IPv6 pueden implementar para ser compatibles con host y enrutadores IPv4.

Los mecanismos de transición son un conjunto de mecanismos y de protocolos implementados en nodos, junto con algunas guías operativas de direccionamiento designadas para hacer la transición a Internet IPv6 con la menor interrupción posible.

Existen dos mecanismos básicos :

- Nivel IP Dual: provee soporte completo para IPv4 e IPv6 en host y enrutadores.

- Túneles IPv6 en IPv4: encapsulan paquetes IPv6 dentro de paquetes IPv4 siendo transportados a través de infraestructura de enrutamiento IPv4. La inexistencia de una infraestructura de enrutamiento IPv6, hace que sea necesario el uso de túneles IPv6 en IPv4 para transportar paquetes IPv6 a través de topologías sólo IPv4, utilizando para ello la infraestructura de encaminamiento IPv4.

En base al mecanismo que utiliza el nodo origen para establecer la dirección del extremo final del túnel se pueden distinguir dos tipos de túneles:

Túneles configurados: En este caso, el nodo origen determina la dirección del final del túnel a través de la información de configuración de la que dispone (p.ej: tablas de enrutamiento). Dicha información debe ser configurada manualmente**.** Este tipo de túneles son unidireccionales; por tanto, para que exista conectividad full-duplex entre dos nodos A y B unidos mediante un túnel configurado se deben establecer sendos túneles en ambos sentidos: es decir, un túnel que tenga como extremo origen el nodo A y como extremo final el nodo B y otro que comience en B y acabe en A.

Túneles automáticos: La dirección IPv4 del otro extremo se obtiene a partir de la dirección IPv6 de destino del paquete a encapsular. Para esta clase de túneles se utilizan direcciones IPv6 del tipo compatible con IPv4, con lo que la dirección IPv4 del extremo final serán los 32 bits de menor orden de la dirección IPv6.

El RFC 1933 (Gilligan & Nordmark, 1996) recoge los mecanismos subyacentes comunes a ambos tipos de túneles, así como una discusión acerca de cómo se determina la dirección del extremo final del túnel en cada uno de los casos.

Dichos mecanismos están diseñados para ser usados por nodos IPv6 que necesitan interoperar con nodos IPv4 y utilizar infraestructuras de enrutamiento IPv4. Se espera que muchos nodos necesitarán compatibilidad por mucho tiempo y quizás indefinidamente. Sin embargo, IPv6 también puede ser usado en ambientes donde no se requiere interoperabilidad con IPv4, así, estos ambientes no necesitan usar ni implementar estos mecanismos.

Se dispone de una gran variedad de mecanismos de entunelamiento, éstos incluyen túneles creados manualmente (RFC 2893), los túneles GRE (Generic Routing Encapsulation), IPv6 en IPv4, túneles automáticos como los túneles compatibles IPv4 y los túneles 6to4.

La transición permitirá usar infraestructuras IPv4 para IPv6 y viceversa, dado que se prevé que su uso será prolongado. Esta es la principal estrategia que se ha de implementar durante el periodo de integración y coexistencia de IPv4 e IPv6 entre proveedores de servicios IPv6 y organizaciones que deseen conectarse a redes IPv6 remotas como 6bone.

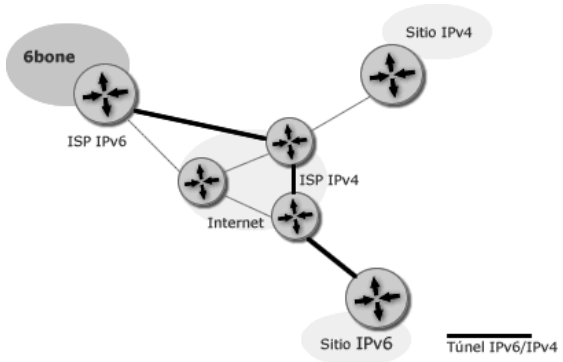


Figura 21 **-** Túneles IPv6 sobre IPv4

Los túneles proporcionan un mecanismo para utilizar las infraestructuras IPv4 mientras la red IPv6 esta siendo implantada. Los extremos finales del túnel siempre son los responsables de realizar la operación de encapsulado de paquetes.

Estos túneles pueden ser utilizados de formas diferentes:

- Enrutador a Enrutador: Enrutadores con doble pila (IPv6/IPv4) se conectan mediante una infraestructura IPv4 y transmiten tráfico IPv6. El túnel comprende un segmento que incluye la ruta completa, extremo a extremo, que siguen los paquetes IPv6.

- Host a Enrutador: Host con doble pila se conectan a un enrutador intermedio (también con doble pila), alcanzable mediante una infraestructura IPv4. El túnel comprende el primer segmento de la ruta seguida por los paquetes.

- Host a Host: Hosts con doble pila interconectados por una infraestructura IPv4. El túnel comprende la ruta completa que siguen los paquetes.

- Enrutador a Host: Enrutadores con doble pila que se conectan a host también con doble pila. El túnel comprende el último segmento de la ruta.

Los túneles se clasifican según el mecanismo por el que el nodo que realiza el encapsulado determina la dirección del nodo extremo del túnel. En los dos primeros casos (Enrutador a Enrutador y host a Enrutador), el paquete IPv6 es tunelizado a un enrutador. El extremo final de este tipo de túnel es un enrutador intermedio que debe desencapsular el paquete IPv6 y reenviarlo a su destino final. En este caso, el extremo final del túnel es distinto del destino final del paquete, por lo que la dirección en el paquete IPv6 no proporciona la dirección IPv4 del extremo final del túnel. La dirección del extremo final del túnel ha de ser determinada a través de información de configuración en el nodo que realiza el túnel. Es lo que se denomina “Túnel configurado”, describiendo aquel tipo de túnel cuyo extremo final es explícitamente configurado. En los otros casos el paquete IPv6 es tunelizado durante todo el recorrido a su nodo destino. El extremo final del túnel es el nodo destino del paquete, y por tanto, la dirección IPv4 está contenida en la IPv6. En este caso se denomina Túnel Automático.

2.15 Delegación de direcciones IPv6

2.15.1 Registros Regionales de Internet

La IANA hizo pública la delegación de un espacio inicial de direcciones IPv6 desde el prefijo 2001::/16 a los Registros Regionales de Internet para comenzar con el desarrollo mundial de

IPv6. En consecuencia, cada uno de estos Registros tiene asignado un prefijo /23 a partir del espacio 2001::/16 así:

- 2001:0200::/23, 2001:0C00::/23 **AP-NIC**: *Asia Pacific - NIC*, Asia Pacífico - Centro de Información de redes, para uso en la zona Asia Pacifico.

- 2001:0400::/23 **ARIN**: *American Registry for Internet Numbers*, Registro Americano de Números de Internet, para uso en América, el Caribe y África.

- 2001:0600::/23, 2001:0800::/23 **RIPE-NIC**: *Reseaux IP Europeennes - NIC*, Redes IP europeas - Centro de información de redes, para uso en Europa.

Nota: Estos Registros Regionales están en capacidad de asignar inicialmente un prefijo /32 a los Proveedores de Servicios de Internet IPv6, y éstos pueden delegar un prefijo /48 a cada Sitio Final.

2.15.2 **6bone: Red Internacional experimental de IPv6 sobre Internet IPv4**

Derivado del proyecto IPv6 del *Internet Engineering Task Force*, Grupo de Ingeniería de Internet (IETF) nace 6bone[25], ésta es una red internacional experimental construida sobre la infraestructura de Internet IPv4 y utilizada para probar el protocolo IPv6 teniendo como fundamento los mecanismos de transición definidos en el RFC 1933, “Mecanismos de transición para enrutadores y hosts IPv6”. La topología de 6bone esta compuesta por “islas“, donde una isla es un conjunto de equipos de cómputo que utilizan el protocolo IPv6 para comunicarse entre sí, unidas por enlaces punto a punto llamados “túneles”, y que opera según el esquema de direccionamiento establecido en el RFC 2471, “Plan de Asignación de direcciones IPv6 para Pruebas”.

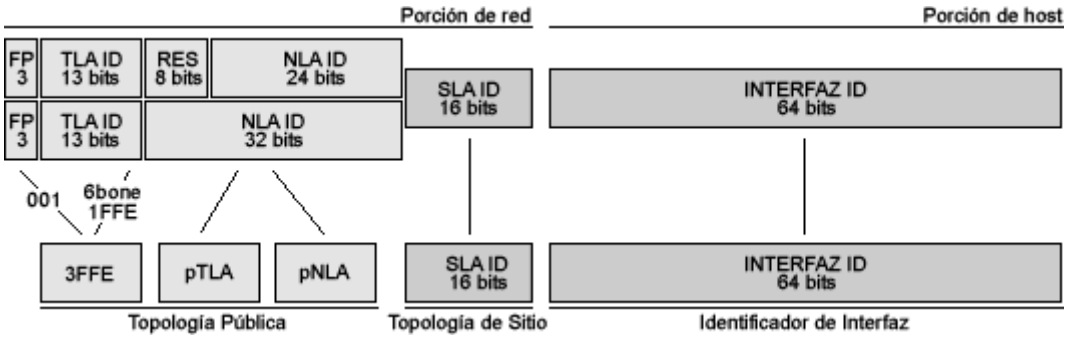


Figura 22 - Estructura de las direcciones de 6bone

[25] 6bone cuenta actualmente con más de 1057 sitios registrados en más 57 países participantes

La IANA asignó a 6bone el Identificador de Agregación de Nivel Superior o TLA ID **1FFE**, que unido al *Format Prefix (FP)* de una dirección Unicast Global Agregable, hace que las direcciones de 6bone inicien por **3FFE**.

En el siguiente nivel de la jerarquía IPv6, el Identificador de Agregación de Nivel Próximo o NLA ID reserva sus primeros bits para crear un pseudo TLA (pTLA) que identifica organizaciones que forman el *backbone* de 6bone. Así, los pTLA's se reconocen por prefijos como **3FFE:nn00::/24**, **3FFE:nnn0::/28** o **3FFE:nnnn::/32**, donde las n's representan el valor asignado a cada pTLA. Los bits restantes del espacio NLA ID pueden ser utilizados por cada pTLA para crear pseudo NLA's (pNLA's) los cuales identifican sitios a quienes se desea dar servicio.

Nota: Con el transcurso del tiempo y según vaya evolucionando el proceso de transición a IPv6, 6bone irá desapareciendo, dando paso de forma transparente a los nuevos Proveedores de Servicios de Internet.

3 Laboratorio Internet IPv6: Red piloto UniCauca IPv6

3.1 Factibilidad

Para conseguir un desarrollo completo del Laboratorio Internet IPv6, son necesarios una serie de recursos descritos a continuación. La total consecución de los objetivos del laboratorio dependerá en buena medida de la disponibilidad de dichos recursos.

3.2 Aprovisionamiento de recursos necesarios

3.2.1 Recursos Hardware

Para implementar el Laboratorio Internet IPv6, se dispone de los PC's y de los dispositivos de internetworking con que cuenta el Laboratorio de Telecomunicaciones de la FIET. Debido a situaciones en el momento inviables como, por ejemplo, la adquisición de enrutadores CISCO con soporte IPv6 para la interconexión de redes, se utilizarán como enrutadores y/o hosts los equipos de cómputo del laboratorio, y en el caso específico, uno de los PC's, hará las veces de enrutador principal del sitio UniCauca IPv6, desde el que se levantará un túnel IPv6 en IPv4 con la UNAM[26] para realizar la conexión a 6bone.

3.2.2 Recursos Software

El software requerido para el desarrollo del proyecto, está bajo la Licencia Pública GNU (GPL) o licencias similares. Esto significa que el software puede ser obtenido sin ningún costo, puede ser instalado en cualquier número de PC's, puede ser copiado o modificado y que no existe restricción en la distribución del software.

3.2.2.1 Selección del Sistema Operativo (S.O)

En la actualidad, existen prototipos e implementaciones de IPv6 para varias plataformas y S.O's: Linux, Windows, Solaris 8, BSDI BSD/OS, OpenBSD, FreeBSD, NetBSD, OpenVMS, Novell Netware6, Mac OS X, entre otros.

Estableciendo criterios de solidez, flexibilidad, seguridad, disponibilidad, y libertad de uso, se ha decido utilizar como plataforma operativa para el desarrollo del proyecto, Red Hat Linux 9 con un kernel estable de la serie 2.4.x. Pero las diferentes distribuciones de Linux (Debian, SuSe, Slackware), habilitan también el soporte IPv6.

[26] Universidad Nacional Autónoma de México

Nota: Se utilizará en las situaciones indicadas el S.O MS Windows 2000 Professional y/o Server con soporte IPv6 “Microsoft IPv6 Technology Preview for Windows 2000”. El Laboratorio de Telecomunicaciones de la FIET facilita la Licencia para éste S.O.

3.2.3 **Red piloto UniCauca IPv6: Espacio de direcciones IPv6 (*bloque* pNLA /48)**

Conforme a lo especificado en el RFC 2471, “Plan de Asignación de direcciones IPv6 para Pruebas”, la UNAM por ser nodo del *backbone* de 6bone ha distribuido su espacio de direcciones IPv6 a partir del prefijo tipo pTLA 3FFE:8070::/28 a comunidades autónomas en el mundo que utilizan la redUNAM IPv6 para pruebas, y dentro de éste espacio ha asignado a UniCauca el bloque pNLA 3FFE:8070:1024::/48.

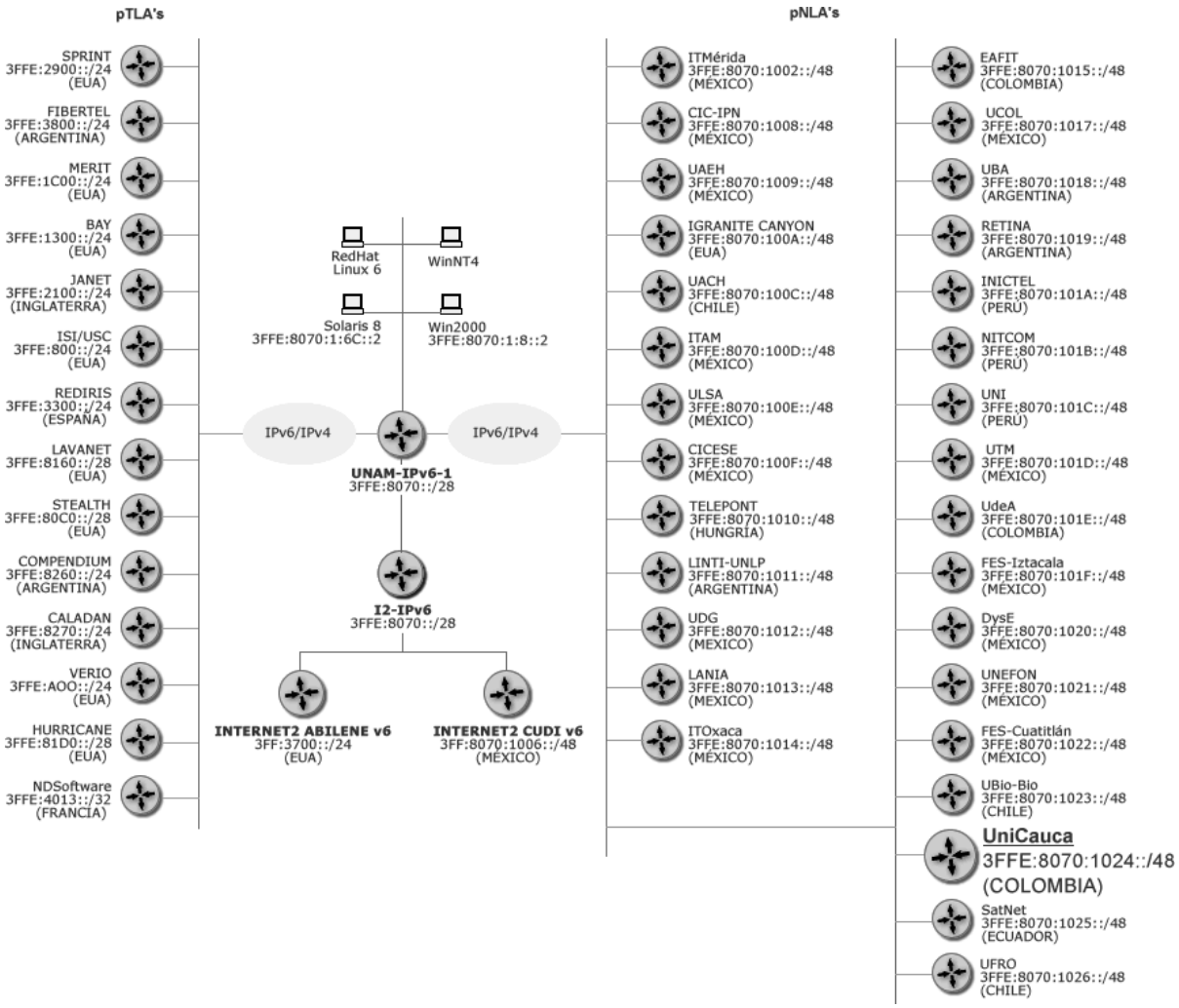


Figura 23 - redUNAM IPv6 para pruebas

En la Figura 24 se aprecia mejor la estructura de las direcciones Globales Agregables asignadas a UniCauca:

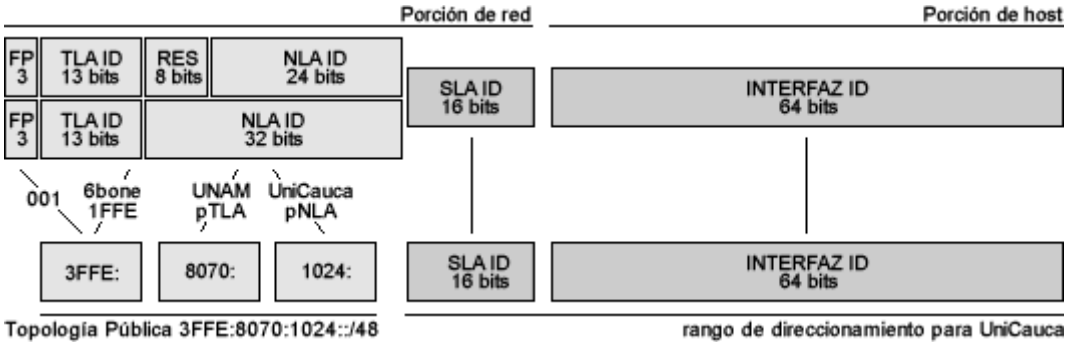


Figura 24 - Estructura de direccionamiento IPv6 para UniCauca

FP: 001; 3bits

TLA ID: 1FFE (6bone); 13bits

NLA ID: pTLA: 8070 (UNAM); 16bits - pNLA: 1024 (UniCauca); 16bits

SLA ID: 16bits

INT ID: 64bits

La Universidad del Cauca tiene capacidad para distribuir internamente diferentes rangos de direcciones IPv6 utilizando el campo SLA ID, el cual a su vez, al igual que se ha hecho con el campo NLA ID, se puede dividir en múltiples niveles.

Estructura de direccionamiento IPv6 para UniCauca utilizando el campo SLA ID

Múltiples niveles (Ejemplo práctico): Dar un prefijo /52 a cada facultad y/o dependencia de UniCauca.

Se dispone así de 2^4 o 16 subredes para facultades y/o dependencias (desde un /48 a un /52; 4bits). De este modo, cada facultad y/o dependencia está en capacidad de direccionar y enrutar 2^{12} o 4096 Sitios Finales (desde un /52 a un /64; 12bits).

3FFE:8070:1024:**0**::/52	Facultad de Ingeniería Electrónica y Telecomunicaciones	Subred **0**
3FFE:8070:1024:**1**::/52	Facultad de Ingeniería Civil	Subred **1**
3FFE:8070:1024:**2**::/52	Facultad de Ciencias de la Salud	Subred **2**
3FFE:8070:1024:**3**::/52	Facultad de Artes	Subred **3**
3FFE:8070:1024:**4**::/52	Facultad de Ciencias Agropecuarias	Subred **4**
3FFE:8070:1024:**5**::/52	Facultad de Ciencias Contables Económicas y Administrativas	Subred **5**
3FFE:8070:1024:**6**::/52	Facultad de Ciencias Humanas y Sociales	Subred **6**
3FFE:8070:1024:**7**::/52	Facultad de Ciencias Naturales, Exactas y de la Educación	Subred **7**
3FFE:8070:1024:**8**::/52	Facultad de Ciencias Políticas y Sociales	Subred **8**
3FFE:8070:1024:**9**::/52	Centro de Educación Abierta y a Distancia	Subred **9**
...	…	**…**
3FFE:8070:1024:**F**::/52	Última Subred	Subred **15**

Cada Sitio Final está en capacidad de asignar 2^{64} o 18,446,744,073,709,551,616 ~ $1.84x10^{19}$ direcciones IPv6 del tipo Global Agregable.

Un único Nivel: Dar un prefijo /64 a UniCauca.

Se dispone así de 2^{16} o 65,536 subredes para UniCauca (desde un /48 a un /64; 16bits)

3FFE:8070:1024:**0000**:/64	Subred **0**
3FFE:8070:1024:**0001**::/64	Subred **1**
...	...
3FFE:8070:1024:**FFFF**::/64	Subred **65,536**

Cada subred está en capacidad de asignar 2^{64} o 18,446,744,073,709,551,616 ~ $1.84x10^{19}$ direcciones IPv6 del tipo Global Agregable.

Nota: La interfaz del enrutador principal de la UNAM que da salida a UniCauca a 6bone, tiene asignada la dirección Global Agregable 3FFE:8070:1024:0000::**1**/64 o 3FFE:8070:1024::**1**/64, así que en la implementación de la red piloto UniCauca IPv6 conviene utilizar la estructura de direccionamiento IPv6 de un único nivel, asignando a la interfaz del enrutador principal del sitio UniCauca IPv6 la dirección Global Agregable 3FFE:8070:1024:0000::**2**/64 o 3FFE:8070:1024::**2**/64.

3.2.4 **Registro del sitio UniCauca en 6bone**

ipv6-site: UNICAUCA
origing: AS64515 **IANA-RSVD2**
descrp: Universidad del Cauca 6Bone pNLA Site, Popayán(Cauca) - Colombia
Proyecto UniCauca IPv6
country: CO - COLOMBIA
prefix: 3FFE:8070:1024::/48
Aggregatable Global Unicast Address
TLA-ID: 0x1ffe, Sub-TLA: 0x100e 6Bone
6BONE:UNAM:UNICAUCA:
contact: FJT1-6BONE
url: http://www.unicauca.edu.co
mnt-by: MNT-UNICAUCA

inet6num: 3FFE:8070:1024::/48
Aggregatable Global Unicast Address
TLA-ID: 0x1ffe, Sub-TLA: 0x100e 6Bone
6BONE:UNAM:UNICAUCA:
netname: UNICAUCA
descr: pNLA delegation for the 6bone
country: CO - COLOMBIA
admin-c FJT1-6BONE
tech-c: FJT1-6BONE
notify: gagredo@unicauca.edu.co
staff_ipv6@ipv6.unam.mx
mnt-by: MNT-UNAM

change: azael@ipv6.unam.mx 17th July 2003

mntner: MNT-UNICAUCA
descr: Maintainer of UNICAUCA 6bone objects
Proyecto UniCauca IPv6
admin-c: FJT1-6BONE
upd-to: dpaz@unicauca.edu.co
auth: CRYPT-PW *
notify: dpaz@unicauca.edu.co
mnt-by: MNT-UNICAUCA
changed: dpaz@unicauca.edu.co 20030717
source: 6BONE

person: Francisco Javier Terán
address: Universidad del Cauca
address: Campus Tulcán / Popayán(Cauca) - Colombia
phone: +5728209800ext.2126
e-mail: fteran@unicauca.edu.co
nic-hdl: FJT1-6BONE
url: http://www.unicauca.edu.co
mnt-by: MNT-UNICAUCA
changed: dpaz@unicauca.edu.co 20030717
source: 6BONE

3.3 Desarrollo técnico

3.3.1 Instalación de Red Hat Linux 9 y de las aplicaciones utilizadas en el Laboratorio Internet IPv6

Para una mayor comprensión y entendimiento de la instalación de RH Linux 9, es conveniente referirse al Manual de instalación de RH Linux 9 para x86: http://www.europe.redhat.com/documentation/rhl9/rhl-ig-x86-es-9/

Entre los varios métodos que pueden utilizarse para instalar RH Linux[27], se prefiere la instalación local desde CD-ROM, ya que el Laboratorio de Telecomunicaciones de la FIET cuenta con los CD-ROM's de Red Hat Linux 9 y los PC's de trabajo permiten iniciar desde una unidad de CD-ROM.

A continuación se muestran los pasos ha seguir durante la instalación de RH Linux 9:

- Bienvenido a Red Hat Linux
- Selección del idioma
- Configuración del teclado

[27]CD-ROM: Si se dispone de un lector de CD-ROM y se tienen los CD's de instalación de RH Linux.
Disco Duro: Si las imágenes ISO de Red Hat Linux, están copiadas en el disco duro local.
NFS: Si la instalación se realiza directamente desde un servidor NFS (Network File System).
FTP: Si la instalación se realiza directamente desde un servidor FTP (File Transfer Protocol).
HTTP: Si la instalación se realiza directamente desde un servidor Web HTTP (HyperText Transmission Protocol).

- Configuración del ratón
- Opciones de instalación: Seleccione qué tipo de instalación desea realizar. Las opciones disponibles son: *Escritorio personal - Estación de trabajo - Servidor - Personalizada*.

  Cualquier tipo de instalación de RH Linux 9, asegura que las aplicaciones y utilidades de red IPv4 e IPv6 esenciales, además del conjunto de archivos de configuración de red y de interfaces, contenidas en los paquetes básicos: *net-tools*, *iputils*, *iproute* e *initscripts*, se instalen por defecto y estén disponibles en el sistema.

  Nota: Los paquetes que RH Linux 9 no instala por defecto, pueden obtenerse utilizando una instalación tipo *Personalizada*, la cual ofrece un control completo sobre la selección de los paquetes software a instalar.

- Particionamiento del sistema
- Configuración del gestor de inicio
- Configuración de red: Cualquier dispositivo de red que tenga el PC y que se desee configurar, será detectado automáticamente por el programa de instalación de RH Linux. De este forma se pueden configurar las tarjetas de red Ethernet en los equipos de cómputo del Laboratorio de Telecomunicaciones de la FIET, teniendo en cuenta la siguiente información:

Intranet UniCauca	
Red	172.16.0.0
Dirección IPv4	172.16.41.**X**[28]
Puerta de Enlace (gateway)	172.16.255.254
Máscara de red	255.255.0.0
DNS Primario	172.16.255.200
DNS Secundario	172.16.255.183
Servidor Proxy	proxy.unicauca.edu.co; puerto: 3128

  Nota: Para cambiar la configuración de red después de la instalación, se utiliza la Herramienta de administración de redes de RH Linux. Para esto, se ejecuta el comando *#redhat-config-network* en una línea de comandos de la shell[29].

- Configuración del Cortafuegos
- Selección del soporte del idioma
- Configuración del huso horario
- Configuración de la contraseña de root
- Configuración de la autenticación

[28] 100 ≤ **X** ≤ 140: rango de direcciones IPv4 asignado al Laboratorio de Telecomunicaciones de la FIET.
[29] Se deben tener privilegios de Administrador del sistema Linux “root”

- Selección de grupos de paquetes: Después de haber elegido una instalación tipo *Personaliza*, se pueden seleccionar grupos de paquetes, paquetes individuales, o una combinación de ambos.

  Se sugiere seleccionar paquetes individuales, así se puede escoger visualizar los paquetes individuales utilizando la *vista árbol*, la cual permite ver los paquetes agrupados según el tipo de aplicación, o la *vista plana*, la cual permite ver todos los paquetes listados en orden alfabético. Entonces se seleccionan los paquetes: *ethereal*, *radvd* y *zebra*, indispensables en éste laboratorio, y que no se instalan por defecto en el S.O.

  Nota: Los paquetes *zebra* y *radvd* se utilizan únicamente en equipos de cómputo que harán las veces de enrutadores IPv6.

- Preparación para la instalación
- Instalación de paquetes
- Creación de un disquete de inicio
- Configuración de la tarjeta de vídeo
- Configuración de X - Control y personalización
- Fin de la instalación

En caso de disponer ya de un S.O RH Linux 9 activo que carezca de los paquetes *ethereal*, *radvd* y *zebra*, los CD-ROM's de RH Linux 9 facilitan éstos paquetes en formato RPM. Para instalar paquetes en formato RPM, es necesario tener privilegios de Administrador del sistema Linux "root" y se debe ejecutar en consola el siguiente comando: *rpm –i nombre-del-paquete-a-instalar.rpm*.

Para los paquetes software que no hacen parte de RH Linux 9, se especificará el sitio en Internet desde el cual se puede descargar una versión de trabajo apropiada, y además se explicará la manera de instalar correctamente éstos paquetes en el S.O.

En los ítem siguientes se clasifican y detallan los paquetes utilizados para un óptimo desarrollo del Laboratorio Internet IPv6.

Nota: Se detallará en las situaciones indicadas la instalación y configuración de aplicaciones adicionales utilizadas en éste Laboratorio.

3.3.1.1 **Aplicaciones Importantes**

Herramientas de configuración IPv6

Paquete net-tools: Contiene las utilidades de networking básicas, utilizadas para controlar las funciones relacionadas con la red en el kernel: *ifconfig*, *route*, *hostname*, *netstat* ...
distribución: Red Hat Linux 9 rpm: net-tools-1.60-12.i386.rpm

Paquete iproute: Contiene utilidades para la configuración de dispositivos de red y para el enrutamiento IP avanzado: *ip*, *iproute2*, *tc ...*
distribución: Red Hat Linux 9 rpm: iproute-2.4.7-7.i386.rpm

Paquete initscripts: Contiene scripts del sistema utilizados en este Laboratorio para configurar interfaces de red: */etc/sysconfig/network-scripts*, */etc/sysconfig/networking* …
distribución: Red Hat Linux 9 rpm: initscripts-7.14-1.i386.rpm

Herramientas de monitoreo IPv6

Paquete iputils: Contiene las utilidades básicas para el monitoreo de redes: *ping*, *ping6*, *tracepath*, *tracepath6*, *traceroute6* ...
distribución: Red Hat Linux 9 rpm: iputils-20020927-2.i386.rpm

Paquete ethereal: Es una utilidad con interfaz gráfica de usuario que permite monitorizar el tráfico en redes.
distribución: Red Hat Linux 9 rpm: ethereal-0.9.8-6.i386.rpm rpm: ethereal-gnome-0.9.8-6.i386.rpm observaciones: Si se desea instalar *ethereal* utilizando el comando *rpm –i nombre-del-paquete-a-instalar.rpm*, es necesario instalar los 2 RPM's anteriores.

3.3.1.2 **Aplicaciones para Enrutamiento**

Únicamente para equipos de cómputo que harán las veces de enrutadores IPv6. Información detallada acerca de los protocolos de enrutamiento ha utilizar, se encuentra en la unidad 4, "Laboratorio Internet IPv6: Enrutamiento en IPv6".

Paquete radvd: Es un demonio de anuncio de rutas en subredes IPv6.
distribución: Red Hat Linux 9 rpm: radvd-0.7.2-2.i386.rpm

Paquete zebra: Gestiona protocolos de enrutamiento basados en TCP/IP. Soporta BGP4, BGP4+, OSPFv2, OSPFv3, RIPv1, RIPv2 y RIPng
distribución: Red Hat Linux 9 rpm: zebra-0.93b-1.i386.rpm

3.3.1.3 **Aplicaciones Adicionales**

Scripts de red IPv6: Se utilizan scripts para inicializar todo lo relacionado con IPv6 y para configurar las direcciones IPv4 e IPv6 de las interfaces. Conviene actualizar a la última versión de los mismos.

Instalación de Scripts de red IPv6 en RH Linux 9

Estos scripts pueden obtenerse en: http://www.bieringer.de/linux/IPv6/IPv6-HOWTO/scripts/current/index.html, aunque RH Linux configura por defecto éstos scripts en la instalación del sistema.

En caso de utilizar la última versión de los scripts de configuración IPv6, se descarga ésta y se descomprime utilizando el comando: *tar -zxvf IPv6-initscripts-version.tar.gz*

Se copian los archivos de script a los directorios correspondientes:

/etc/sysconfig/network-scripts/network-functions-ipv6
/etc/sysconfig/network-scripts/init.ipv6-global
/etc/sysconfig/network-scripts/ifup-ipv6
/etc/sysconfig/network-scripts/ifdown-ipv6
/etc/sysconfig/network-scripts/ifup-sit
/etc/sysconfig/network-scripts/ifdown-sit

/etc/ppp/ip-up.ipv6to4
/etc/ppp/ip-down.ipv6to4
/etc/ppp/ipv6-up
/etc/ppp/ipv6-down

/usr/sbin/test-ipv6-installation
/etc/sysconfig/static-routes-ipv6

En el archivo *sysconfig-ipv6.txt* que viene con el paquete de scripts, se da información detallada de los parámetros que se pueden configurar en cada script. Para comprobar que la configuración es correcta se puede ejecutar el script: */usr/sbin/test-ipv6-installation* que viene con el paquete.

Existen muchas aplicaciones que funcionan con IPv6[30]. Las últimas versiones de los servidores más usados para los servicios básicos ya soportan IPv6:

- Web (Apache: http://www.apache.org)
- DNS (BIND: http://www.isc.org)
- FTP
- Telnet
- SSH (OpenSSH: http://www.openssh.com)
- E-mail (Sendmail: http://www.sendmail.org)

También existen clientes de éstos servicios con soporte IPv6.

3.3.2 **Soporte IPv6 en RH Linux**

Red Hat Linux soporta IPv6 a partir de su versión 7.1, y se implementa utilizando el módulo IPv6 que se incorpora de manera dinámica al kernel de Linux, o compilando el kernel para ofrecer soporte nativo a IPv6.

3.3.2.1 **Soporte IPv6 en RH Linux utilizando el módulo IPv6**

Linux, implementa IPv6 como un módulo del kernel a partir de las versiones 2.2.x y 2.4.x, por lo que normalmente se instala el módulo IPv6 y se carga de manera automática durante el inicio del sistema.

Estos son los pasos que permiten habilitar el módulo IPv6 en RH Linux:

Examinar el soporte IPv6 en el kernel de Linux: Para asegurarse que el kernel que se está corriendo soporte IPv6, se comprueba que exista la entrada */proc/net/if_inet6* en el sistema de archivos /proc.

Cargar el módulo IPv6 en el kernel de Linux: En caso de que la entrada */proc/net/if_inet6* no exista, se carga el módulo IPv6 utilizando el comando *modprobe ipv6.* Si esto es exitoso, el módulo *ipv6* será listado en pantalla al ejecutar el comando *lsmod*.

Cargar el módulo IPv6 automáticamente: Para esto se añade al archivo */etc/modules.conf* ó */etc/conf.modules* las líneas:

[30] Para una información más detallada visitar: http://www.bieringer.de/linux/IPv6/status/IPv6+Linux-status-apps.html

```
alias net-pf-10 ipv6
alias sit0 ipv6
alias sit1 ipv6
alias tun6to4 ipv6
```

También es posible deshabilitar la carga automática del módulo usando la línea:

```
alias net-pf-10 off
```

3.3.2.2 **Compilar el kernel para el soporte IPv6 en RH Linux**

Si los resultados anteriores son negativos ó si se desea tener soporte IPv6 nativo en el kernel, se tienen las siguientes alternativas:

- Compilar / recompilar las fuentes del kernel para habilitar el soporte de IPv6.
- Actualizar la distribución a una que soporte las capacidades de IPv6[31].
- Compilar un nuevo kernel utilizando las extensiones del Proyecto USAGI[32], las cuales proveen paquetes precompilados y parches para dar soporte a IPv6.

A continuación se describen de forma general los aspectos principales que dan soporte IPv6 nativo en RH Linux, mediante la compilación e instalación de un nuevo kernel[33]:

- Ingresar al directorio de fuentes: *#cd /usr/src* y remover el enlace simbólico al directorio actual de fuentes del kernel: *#rm linux*

- Obtener y descomprimir la última versión estable de las fuentes del kernel: http://www.kernel.org y ejecutar el comando: *#tar -zxvf* ***ruta-nuevas-fuentes****/linux-****version****.tar.gz –C /usr/src*

- Renombrar el nuevo directorio de fuentes, para una mejor identificación:
  *#mv linux linux-****version***

- Crear un enlace simbólico al directorio en cuestión: *#ln -s linux-****version*** *linux*

31 Información detallada sobre el soporte IPv6 en las distribuciones Linux más comunes puede encontrarse en "IPv6 Linux-Status-Distribution": http://www.bieringer.de/linux/IPv6/status/IPv6+Linux-status-distributions.html

32 Proyecto USAGI: El cual es un esfuerzo de empresas japonesas para implementar todo el stack del protocolo IPv6 en Linux - http://www.linux-ipv6.org

33 Se recomienda ejecutar los comandos *#/sbin/lspci* y *#cat /proc/cpuinfo*, para obtener información detallada acerca de las características HW del PC. En todo caso es adecuado utilzar un Kernel-HOWTO: www.tldp.org/HOWTO/Kernel-HOWTO.html

- Ingresar al directorio de fuentes del kernel: *#cd linux*

- Configurar el kernel para el soporte de IPv6:
  #make xconfig ó #make menuconfig

Grupo	Opción	Selección
Code maturuty level options	Prompt for development and/or incomplete code/drivers	[*]
Loadable module support	Enable loadable module support	[*]
	Set version information on all module symbols	[*]
	Kernel module loader	[*]
Networking options	Packet socket	[*]
	Unix domain sockets	[*]
	TCP/IP networking	[*]
	The IPv6 protocol (EXPERIMENTAL)	[*]

Tabla 4 - Opciones de kernel específicas para IPv6

Compilación e Instalación

- Realizar una copia de seguridad del archivo de configuración del kernel: *#cp .config ../config-**version** (opcional)*

- Modificar la etiqueta de EXTRAVERSION, para crear una nueva etiqueta de versión (una sub-versión), editando el Makefile: *#vi Makefile EXTRAVERSION = -IPV6*

- Compilar el nuevo kernel: *#make dep*; *#make clean*; *#make* {*zImage* | *bzImage*};

- Compilar los módulos: *#make modules*; *#make modules_install*

- Crear una imagen inicial RAMDisk: *#mkinitrd /boot/initrd-**version**.img version*

- Remover el enlace simbólico al mapa del kernel, System.map: *#rm /boot/System.map*

- Copiar y renombrar el nuevo kernel y el System.map:
  #cp arch/i386/boot/{*zImage* | *bzImage*} */boot/vmlinuz-**version***
  *#cp System.map /boot/System.map-**version***

- Crear un enlace simbólico al *System.map-**version***: *#ln -s /boot/System.map-**version** /boot/System.map*

- Editar */etc/lilo.conf* o */etc/grub.conf* y añadir la entrada al nuevo kernel

- Reiniciar el sistema y seleccionar el nuevo kernel: *#reboot*

3.3.3 **Configuración IPv6 de red en RH Linux**

Una vez instaladas las aplicaciones y el soporte IPv6 en los PC's con S.O RH Linux 9 que forman la red piloto UniCauca IPv6, se está en capacidad de configurar interfaces de red y de interoperar directamente con nodos IPv6. Para esto se trabajará directamente sobre los Scripts de Red instalados en el sistema, los cuales proveen un control permanente de las interfaces de red a utilizar; de este modo, cuando RH Linux inicia, utiliza estos archivos para saber que interfaces utilizar automáticamente y como configurarlas para que operen correctamente.

A continuación se explica la manera de configurar los parámetros de red IPv6 en los equipos de cómputo del Laboratorio de Telecomunicaciones de la FIET, los cuales utilizarán direcciones Globales Agregables, a partir del prefijo tipo pNLA 3FFE:8070:1024::/48 asignado a UniCauca.

3.3.3.1 **A tener en cuenta**

- Para cambiar el nombre a un PC se agrega en */etc/sysconfig/network*, la línea:

```
HOSTNAME=nombre-del-equipo
```

- El nombre del equipo puede verse en */proc/sys/kernel/hostname*, o simplemente ejecutando el comando *hostname* sin ningún parámetro.

- Se deben añadir entradas en */etc/hosts* para IPv6:

```
::1             localhost               ip6-localhost           ip6-loopback
FE00::0         ip6-localnet
FF00::0         ip6-mcastprefix
FF02::1         ip6-allnodes
FF02::2         ip6-allrouters
FF02::3         ip6-allhosts
```

- Comprobar que en */etc/protocols* aparece:

```
ipv6            41              IPv6
ipv6-route      43              IPv6-Route
ipv6-frag       44              IPv6-Frag
ipv6-crypt      50              IPv6-Crypt
ipv6-auth       51              IPv6-Auth
ipv6-icmp       58              IPv6-ICMP
ipv6-nonxt      59              IPv6-NoNxt
ipv6-opts       60              IPv6-Opts
```

3.3.3.2 **Asignación de direcciones IPv6 a interfaces de red**

Si se ejecuta en la consola de Linux, el comando *ifconfig* se obtiene la siguiente salida:

```
[root@ipv6 /]# ifconfig

eth0      Link encap:Ethernet  HWaddr 00:B0:D0:C2:D7:76
          inet addr:172.16.41.X  Bcast:172.16.255.255  Mask:255.255.0.0
          inet6 addr: fe80::2b0:d0ff:fec2:d776/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:7728540 errors:10 dropped:0 overruns:51 frame:17
          TX packets:10007 errors:0 dropped:0 overruns:0 carrier:0
          collisions:140 txqueuelen:100
          RX bytes:3041051657 (2900.1 Mb)  TX bytes:915038 (893.5 Kb)
          Interrupt:5 Base address:0xe880

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:70976 errors:0 dropped:0 overruns:0 frame:0
          TX packets:70976 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:4848248 (4.6 Mb)  TX bytes:4848248 (4.6 Mb)
```

Como se puede ver, la interfaz Ethernet eth0 además de disponer de una dirección IPv4, tiene una nueva dirección IPv6 (inet6 addr). Junto a la dirección, aparece su ámbito (Scope:Link), es decir, Local de Enlace, ya que comienza por el *Format Prefix (FP)* FE80/10 que es precisamente el prefijo utilizado por las direcciones Link-Local. El soporte IPv6 en RH Linux habilita el formato de token EUI-64[34] y la dirección Local de Enlace se configura automáticamente al iniciar el S.O. También se puede ver que la interfaz de Loopback ya está configurada (inet6 addr: ::1/128 Scope:Host).

El proceso que se utiliza para convertir una dirección Ethernet al formato de token EUI-64 se ilustra en la Figura 25. Debido a que una dirección Ethernet se compone de 6bytes (48bits), para que el token resultante tenga una longitud de 64bits se insertan 2 bytes más (FF y FE) entre los bytes tercero y cuarto de dicha dirección.

A continuación se complementa el bit U/L (Universal/Local). Dicho bit es el siguiente bit al de más bajo orden del primer byte e indica si la dirección es globalmente única (1) o no (0).

[34] La dirección IPv6 Local de Enlace que aparece al ejecutar ifconfig, es el resultado de concatenar al Format Prefix (FP) FE80/10 el token que se obtiene al aplicar al Identificador de interfaz (dirección Ethernet) de la tarjeta de red el formato EUI-64.

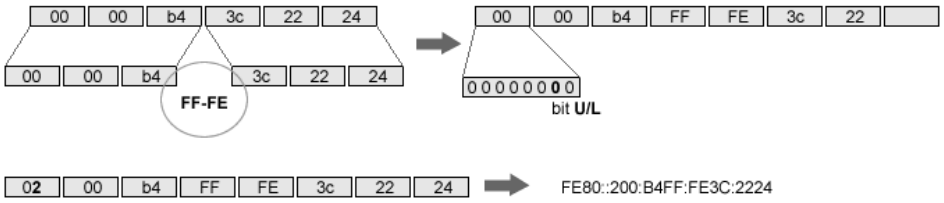


Figura 25 - Proceso de creación de la dirección Local de Enlace

Asignación de direcciones Globales Agregables a Hosts

Se utilizará el mecanismo de *autoconfiguración sin control de estado*, el cual permite asignar una dirección Global Agregable en forma autónoma y automática, a partir del Identificador de Interfaz y del prefijo anunciado por el enrutador IPv6 del segmento de red. Para esto se trabaja directamente sobre los archivos de configuración de interfaz:

- El archivo */etc/sysconfig/network* debe tener respecto a IPv6 las siguientes entradas:

```
NETWORKING_IPV6=yes
# NETWORKING_IPV6=<indicador>, donde <indicador> es uno de los siguientes valores:
# yes:    Habilita la inicialización Global de IPv6
# no:     Deshabilita la inicialización Global de IPv6

IPV6_AUTOCONF=yes
# IPV6_AUTOCONF=<indicador>, donde <indicador> es uno de los siguientes valores:
# yes:    Habilita la autoconfiguración de direcciones IPv6
# no:     Deshabilita la autoconfiguración de direcciones IPv6

IPV6FORWARDING=no
# IPV6FORWARDING=<indicador>, donde <indicador> es uno de los siguientes valores:
# yes:    Habilita el reenvío de paquetes IPv6
# no:     Deshabilita el reenvío de paquetes IPv6

IPV6_AUTOTUNEL=no
# IPV6_AUTOTUNEL=<indicador>, donde <indicador> es uno de los siguientes valores:
# yes:    Habilita la inicialización de túneles automáticos
# no:     Deshabilita la inicialización de túneles automáticos
```

En el directorio */etc/sysconfig/network-scripts/* se edita el archivo de configuración de interfaz apropiado (Ejemplo: ifcfg-eth0), y se agregan las siguiente entradas:

```
IPV6INIT=yes
# IPV6INIT=<indicador>, donde <indicador> es uno de los siguientes valores:
# yes:    Habilita la inicialización de IPv6 en ésta interfaz
# no:     Deshabilita la inicialización de IPv6 en ésta interfaz

IPV6_ROUTER=no
# IPV6_ROUTER=<indicador>, donde <indicador> es uno de los siguientes valores:
# yes:    Habilita el envío de Anuncios de enrutamiento en ésta interfaz
# no:     Deshabilita el envío de Anuncios de enrutamiento en ésta Interfaz
```

Asegúrese que los archivos de configuración de interfaz localizados en los directorios */etc/sysconfig/network-scripts/* y */etc/sysconfig/networking/devices/* tengan iguales entradas.

Nota: La *autoconfiguración sin control de estado* se basa en la siguiente premisa: En la red se encuentra un enrutador con el *Router Advertisements*, Anuncio de enrutamiento (RA) activado.

Asignación de direcciones Globales Agregables a Enrutadores

En este caso conviene establecer un mecanismo de asignación de direcciones IPv6 estático, entonces se procede de la siguiente manera:

- El archivo */etc/sysconfig/network* debe tener respecto a IPv6 las siguientes entradas:

```
NETWORKING_IPV6=yes
# NETWORKING_IPV6=<indicador>, donde <indicador> es uno de los siguientes valores:
# yes:    Habilita la inicialización Global de IPv6
# no:     Deshabilita la inicialización Global de IPv6

IPV6_AUTOCONF=no
# IPV6_AUTOCONF=<indicador>, donde <indicador> es uno de los siguientes valores:
# yes:    Habilita la autoconfiguración de direcciones IPv6
# no:     Deshabilita la autoconfiguración de direcciones IPv6

IPV6FORWARDING=yes
# IPV6FORWARDING=<indicador>, donde <indicador> es uno de los siguientes valores:
# yes:    Habilita el reenvío de paquetes IPv6
# no:     Deshabilita el reenvío de paquetes IPv6

IPV6_AUTOTUNEL=no
# IPV6_AUTOTUNEL=<indicador>, donde <indicador> es uno de los siguientes valores:
# yes:    Habilita la inicialización de túneles automáticos
# no:     Deshabilita la inicialización de túneles automáticos
```

En el directorio */etc/sysconfig/network-scripts/* se edita el fichero de configuración de interfaz apropiado (Ejemplo: ifcfg-eth0), y se agregan las siguiente entradas:

```
IPV6INIT=yes
# IPV6INIT=<indicador>, donde <indicador> es uno de los siguientes valores:
# yes:    Habilita la inicialización de IPv6 en ésta interfaz
# no:     Deshabilita la inicialización de IPv6 en ésta interfaz

IPV6_ROUTER=yes
# IPV6_ROUTER=<indicador>, donde <indicador> es uno de los siguientes valores:
# yes:    Habilita el envío de Anuncios de enrutamiento en ésta interfaz
# no:     Deshabilita el envío de Anuncios de enrutamiento en ésta interfaz

IPV6ADDR=3FFE:8070:1024:1::10/64
# IPV6ADDR=<dir-IPv6>/<long-prefijo>, donde <dir-IPv6>/<long-prefijo> es la dirección y el prefijo de red IPv6 asignados a la Interfaz
```

Asegúrese que los archivos de configuración de interfaz localizados en los directorios */etc/sysconfig/network-scripts/* y */etc/sysconfig/networking/devices/* tengan iguales entradas.

Finalmente para trabajar con los nuevos parámetros de red configurados, se reinicia el servicio de red ejecutando el comando #*service network restart*.

3.3.3.3 **Comandos IPv6 útiles en RH Linux**

Muchas veces es necesario realizar un control ágil de los dispositivos de networking conectados al sistema, por lo que resultar más fácil trabajar con comandos ejecutados en la consola de Linux, que trabajar directamente sobre los Scripts de Red instalados en el S.O.

A continuación se muestra una lista de comandos útiles que puede ayudar en el desarrollo del Laboratorio Internet IPv6:

- Mostrar direcciones IPv6: Se puede hacer mediante el uso de los comandos *ip o ifconfig.*

/sbin/ip - 6 addr show dev <interfaz>
/sbin/ifconfig <interfaz>, donde <*interfaz*> puede ser: lo, eth0 ...

Ejemplo: */sbin/ip - 6 addr show dev eth0, /sbin/ifconfig eth0*

- Añadir una dirección IPv6: Se puede hacer mediante el uso de los comandos *ip o ifconfig.*

/sbin/ip -6 addr add <dir-IPv6>/<long-prefijo> dev <interfaz>
/sbin/ifconfig <interfaz> inet6 add <dir-IPv6>/<long-prefijo>

Ejemplo: */sbin/ip -6 addr add 3FFE:8070:1024:1::10/64 dev eth0,*
/sbin/ifconfig eth0 inet6 add 3FFE:8070:1024:1::10/64

- Eliminar una dirección IPv6: Se puede hacer mediante el uso de los comandos *ip o ifconfig.*

/sbin/ip -6 addr del <dir-IPv6>/<long-prefijo> dev <interfaz>
/sbin/ifconfig <interfaz> inet6 del <dir-IPv6>/<long-prefijo>

Ejemplo: */sbin/ip -6 addr del 3FFE:8070:1024:1::10/64 dev eth0,*
/sbin/ifconfig eth0 inet6 del 3FFE:8070:1024:1::10/64

- Mostrar rutas IPv6: Se puede hacer mediante el uso de los comandos *ip o route.*

/sbin/ip - 6 route show dev <interfaz>
/sbin/route -A inet6

Ejemplo: */sbin/ip -6 route show dev eth0, /sbin/route -A inet6* | *grep -w "eth0"*

- Añadir una ruta IPv6 accesible a través de una gateway: Se puede hacer mediante el uso de los comandos *ip o route.*

/sbin/ip - 6 route add <red-IPv6>/<long-prefijo> via <dir-IPv6> dev <interfaz>
/sbin/route -A inet6 add <red-IPv6>/<long-prefijo> gw <dir-IPv6> dev <interfaz>

Ejemplo: */sbin/ip - 6 route add 3FFE:8070:1024:1::/64 via 3FFE:8070:1024:1::1 dev eth0,*
/sbin/route -A inet6 add 3FFE:8070:1024:1::/64 gw 3FFE:8070:1024:1::1 dev eth0

- Eliminar una ruta IPv6 accesible a través de una gateway: Se puede hacer mediante el uso de los comandos *ip o route.*

/sbin/ip - 6 route del <red-IPv6>/<long-prefijo> via <dir-IPv6> dev <interfaz>
/sbin/route -A inet6 del <red-IPv6>/<long-prefijo> gw <dir-IPv6> dev <interfaz>

Ejemplo: */sbin/ip - 6 route del 3FFE:8070:1024:1::/64 via 3FFE:8070:1024:1::1 dev eth0,*
/sbin/route -A inet6 del 3FFE:8070:1024:1::/64 gw 3FFE:8070:1024:1::1 dev eth0

- Añadir una ruta IPv6 accesible a través de una interfaz: Se puede hacer mediante el uso de los comandos *ip o route.*

/sbin/ip - 6 route add <red-IPv6>/<long-prefijo> dev <interfaz> metric 1
/sbin/route -A inet6 add <red-IPv6>/<long-prefijo> dev <interfaz>

Ejemplo: */sbin/ip - 6 route add 3FFE:8070:1024:1::/64 dev eth0 metric 1*,
/sbin/route -A inet6 add 3FFE:8070:1024:1::/64 dev eth0

- Eliminar una ruta IPv6 accesible a través de una interfaz: Se puede hacer mediante el uso de los comandos *ip o route.*

/sbin/ip - 6 route del <red-IPv6>/<long-prefijo> dev <interfaz> metric 1
/sbin/route -A inet6 del <red-IPv6>/<long-prefijo> dev <interfaz>

Ejemplo: */sbin/ip - 6 route del 3FFE:8070:1024:1::/64 dev eth0 metric 1,*
/sbin/route -A inet6 del 3FFE:8070:1024:1::/64 dev eth0

3.3.4 **Soporte IPv6 en MS Windows 2000**

A partir de las versiones de los S.O's MS Windows XP y MS Windows 2003 Server el stack de protocolos IPv6 viene preinstalado y su configuración es muy sencilla.[35]

Para el desarrollo del Laboratorio Internet IPv6, se utilizará en las situaciones indicadas el S.O MS Windows 2000 Professional y/o Server[36], y de esta manera se probará la interoperabilidad del protocolo IPv6 en más de un S.O. Es de señalar que el S.O MS Windows 2000 no tiene preinstalado el stack de protocolos IPv6, así que es necesario habilitar el soporte IPv6 utilizando las extensiones "Microsoft IPv6 Technology Preview for Windows 2000".

Para tener soporte IPv6 en MS Windows 2000 se recomienda hacer lo siguiente:

- Se puede utilizar la interfaz gráfica de Windows para verificar el S.O, su versión y el nivel de Service Pack (S.P) instalado. Para esto, se selecciona *Propiedades* dando click derecho en el icono "Mi PC" del Escritorio, y luego, en la pestaña General de las Propiedades del Sistema, se lee la información que se muestra debajo del ítem Sistema.

- Dependiendo del nivel de S.P instalado, se procede según corresponda: MS Windows 2000 con nivel de S.P 1, 2, 3 o 4.

3.3.4.1 **Instalación del soporte IPv6 en MS Windows 2000**

- Descargue el archivo *tpipv6-001205.exe* desde:
  http://msdn.microsoft.com/downloads/sdks/platform/tpipv6/download.asp

- Ejecute el archivo *tpipv6-001205.exe* y extraiga su contenido en una carpeta local (Ejemplo: C:\IPv6Kit).

[35] En el sitio http://www.microsoft.com/ipv6 aparece todo el soporte que provee Microsoft al protocolo IPv6 en sus S.O's.
[36] Esto debido a que el Laboratorio de Telecomunicaciones de la FIET dispone de Licencias para éste S.O.

- Ejecute desde C:\IPv6Kit, utilizando el interprete de comandos de Windows, el comando *setup.exe -x*. Después extraiga los archivos a una subcarpeta en la carpeta actual (Ejemplo: C:\IPv6Kit\ipv6).

- Desde C:\IPv6Kit\ipv6, acceda al archivo *hotfix.inf* utilizando un editor de texto.

- En la sección [Version] del archivo *hotfix.inf*, modifique la línea NtServicePackVersion según corresponda, posteriormente guarde los cambios:

	SP1	**SP2**	**SP3**	**SP4**
NtServicePackVersion	256	512	768	1024

- Desde C:\IPv6Kit\ipv6, ejecute el archivo *hotfix.exe*. Probablemente en este punto el S.O se reinicie.

3.3.4.2 **Habilitación del soporte IPv6 en MS Windows 2000**

- Desde el Escritorio, dando click derecho en el icono “Entorno de Red”, seleccione *Propiedades*. Nuevamente dando click derecho sobre el dispositivo de red en el que se quiere habilitar IPv6, de click en *Propiedades*.

- Haciendo click sobre *Instalar*, seleccione *Protocolo* y *Agregar*. Ahora seleccione “Microsoft IPv6 Protocol” y luego OK.

- Para probar que el stack de Protocolos IPv6 ha sido correctamente instalado y habilitado, se ejecuta en una consola de Windows el comando *ipv6 if*, el cual muestra la configuración y las direcciones IPv6 autoconfiguradas[37] para cada interfaz de red existente.

```
Interface 5 (site 1): Conexión de Area local 2
  uses Neighbor Discovery
  link-level address: 00-03-ce-89-2f-c0
    preferred address fe80::203:ceff:fe89:2fc0, infinite/infinite
    multicast address ff02::1, 1 refs, not reportable
    multicast address ff02::1:ff89:2fc0, 1 refs, last reporter
  link MTU 1500 (true link MTU 1500)
  current hop limit 128
  reachable time 32500ms (base 30000ms)
  retransmission interval 1000ms
  DAD transmits 1

Interface 4 (site 1): 6-over-4 Virtual Interface
  uses Neighbor Discovery
  link-level address: 172.16.41.113
    preferred address fe80::ac10:2971, infinite/infinite
```

[37] La *autoconfiguración sin control de estado* se basa en la siguiente premisa: En la red se encuentra un enrutador con el *Router Advertisements*, Anuncio de enrutamiento (RA) activado.

```
   multicast address ff02::1, 1 refs, not reportable
   multicast address ff02::1:ff10:2971, 1 refs, last reporter
  link MTU 1280 (true link MTU 65515)
  current hop limit 128
  reachable time 33000ms (base 30000ms)
  retransmission interval 1000ms
  DAD transmits 1

Interface 3 (site 1): Conexión de Area local
  uses Neighbor Discovery
  link-level address: 00-b0-d0-d9-98-7c
    preferred address fe80::2b0:d0ff:fed9:987c, infinite/infinite
    multicast address ff02::1, 1 refs, not reportable
    multicast address ff02::1:ffd9:987c, 1 refs, last reporter
  link MTU 1500 (true link MTU 1500)
  current hop limit 128
  reachable time 22500ms (base 30000ms)
  retransmission interval 1000ms
  DAD transmits 1

Interface 2 (site 0): Tunnel Pseudo-Interface
  does not use Neighbor Discovery
  link-level address: 0.0.0.0
    preferred address ::172.16.41.113, infinite/infinite
  link MTU 1280 (true link MTU 65515)
  current hop limit 128
  reachable time 0ms (base 0ms)
  retransmission interval 0ms
  DAD transmits 0

Interface 1 (site 0): Loopback Pseudo-Interface
  does not use Neighbor Discovery
  link-level address:
    preferred address ::1, infinite/infinite
  link MTU 1500 (true link MTU 1500)
  current hop limit 1
  reachable time 0ms (base 0ms)
  retransmission interval 0ms
  DAD transmits 0
```

3.4 Conexión con el Exterior - 6bone

La inexistencia de una infraestructura de enrutamiento IPv6, hace que sea necesario el uso de túneles IPv6 en IPv4 para transportar paquetes IPv6 a través de topologías sólo IPv4, utilizando para ello la infraestructura de encaminamiento IPv4.

3.4.1 Túnel con la Universidad Nacional Autónoma de México

La UNAM ha realizado diversas experiencias en torno a IPv6, cuenta con una red IPv6 para producción que ofrece servicios de Internet basados en IPv6 para usuarios en México y Latinoamérica y una red IPv6 para pruebas internacionales, que permite configurar túneles y delegar espacio de direcciones IPv6. La colaboración de la UNAM, para el acceso a 6bone de UniCauca, permite establecer un túnel IPv6 en IPv4 punto a punto entre estas dos instituciones.

3.4.2 Túneles IPv6 en IPv4 en RH Linux

En Linux éstos túneles vienen representados por las interfaces de encapsulación **sit***n.* Así, la interfaz sit0 es la interfaz para túneles automáticos, mientras que el resto de las interfaces sit (sit1, sit2, ...) se utilizan para establecer túneles IPv6 en IPv4 manualmente configurados.

Los túneles configurados son de sentido único, es decir, que para establecer una conexión full-duplex es necesario establecer túneles configurados en ambos sentidos. Por tal razón, la UNAM debe configurar un túnel simétrico al configurado por UniCauca.

3.4.3 Establecimiento de un túnel IPv6 en IPv4 entre UniCauca y la UNAM

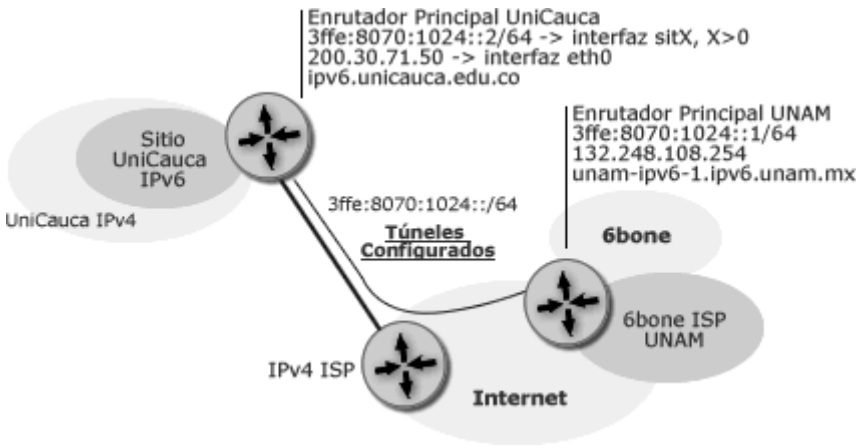


Figura 26 - Topología de túneles estáticos entre los sitios UniCauca IPv6 y UNAM IPv6

A continuación, se muestran los datos de los equipos entre los que se deben establecer los túneles:

	UniCauca[38]	**UNAM**
Hostname	ipv6.unicauca.edu.co	unam-ipv6-1.ipv6.unam.mx
Dir. IPv4 Oficial	200.30.71.50 - Interfaz **eth0**	132.248.108.254
Dir. IPv6 Global Agregable	3FFE:8070:1024::**2**/64 - Interfaz **sitX**[39]	3FFE:8070:1024::**1**/64

Nota: Cada una de las tareas presentadas a continuación se identifica como requerida para la configuración básica de un túnel manualmente configurado entre UniCauca y la UNAM.

Tareas:

- Habilitar el soporte IPv6 en el enrutador principal del sitio UniCauca IPv6
- Configurar interfaces IPv4 en el enrutador principal del sitio UniCauca IPv6
- Configurar una interfaz de túnel sitX en el enrutador principal del sitio UniCauca IPv6

Habilitar el soporte IPv6 en el enrutador principal del sitio UniCauca IPv6

Nota: Referirse a la Sección 3.3.2: Soporte IPv6 en RH Linux

Configuración de interfaces IPv4 en el enrutador principal del sitio UniCauca IPv6

Para configurar interfaces IPv4, se utiliza la Herramienta de administración de redes de RH Linux. Para esto, se ejecuta el comando *redhat-config-network* en una línea de comandos de la shell[40].

Información a tener en cuenta:[41]

hostname: ipv6.unicauca.edu.co		
	Internet	**Intranet**
Interfaz Ethernet	eth0	eth1
Red	200.30.71.0	172.16.0.0
Dirección IPv4	200.30.71.50	172.16.41.108
Puerta de Enlace (gateway)	200.30.71.254	-
Máscara de red	255.255.255.0	255.255.0.0
Servidor de Nombres de Dominio		
DNS Primario	172.16.255.200	
DNS Secundario	172.16.255.183	
DNS Terciario	200.30.71.129	

Es conveniente reiniciar el servicio de red ejecutando el comando *#service network restart*, para actualizar los nuevos cambios.

[38] Enrutador principal del sitio UniCauca IPv6
[39] sitX (X>0): Túnel configurado IPv6 en IPv4
[40] Se deben tener privilegios de Administrador del sistema Linux "root"
[41] A excepción de la dirección oficial IPv4: 200.30.71.50 asignada al equipo ipv6.unicauca.edu.co, los valores restantes están sujetos a cambios, así que se hace necesario consultar información actualizada en la Red de Datos de la Universidad del Cauca.

Para verificar la Tabla de Enrutamiento IPv4 actual del S.O, se ejecuta el comando *route –n.* Se debe obtener la siguiente salida[42]:

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
200.30.71.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
172.16.0.0	0.0.0.0	255.255.0.0	U	0	0	0	eth1
169.254.0.0	0.0.0.0	255.255.0.0	U	0	0	0	eth1
127.0.0.0	255.0.0.0	0.0.0.0	U	0	0	0	lo
0.0.0.0	200.30.71.254	0.0.0.0	UG	0	0	0	eth0

Nota: En el archivo */ets/sysconfig/network* comprobar que exista la entrada: GATEWAY=200.30.71.254

A continuación se muestra una lista de comandos útiles que se pueden utilizar para configurar rutas estáticas en la tabla de enrutamiento IPv4 del kernel de Linux, haciendo uso del comando *route*:

- Añadir una ruta IPv4 accesible a través de una gateway: ruta por defecto (Internet) a través de la gateway 200.30.71.254

/sbin/route add –net <red-IPv4> gw <dir-IPv4>

Ejemplo: */sbin/route add -net default gw 200.30.71.254*

- Eliminar una ruta IPv4 accesible a través de una gateway:

/sbin/route del –net <red-IPv4> gw <dir-IPv4>

Ejemplo: */sbin/route del -net default gw 200.30.71.254*

- Añadir una ruta IPv4 accesible a través de una interfaz: ruta a la red 200.30.71.0 a través de la interfaz **eth0 -** ruta a la red 172.16.0.0 a través de la interfaz **eth1**

/sbin/route add –net <red-IPv4> netmask <dir-IPv4> dev <interfaz>

Ejemplos: */sbin/route add -net 200.30.71.0 netmask 255.255.255.0 dev eth0*
/sbin/route add -net 172.16.0.0 netmask 255.255.0.0 dev eth1

- Eliminar una ruta IPv4 accesible a través de una interfaz:

[42] Red Hat Linux 9, añade por defecto la red 169.254.0.0

/sbin/route del –net <red-IPv4> netmask <dir-IPv4> dev <interfaz>

Ejemplos: /sbin/route del -net 200.30.71.0 netmask 255.255.255.0 dev eth0
/sbin/route del -net 172.16.0.0 netmask 255.255.0.0 dev eth1

Configuración de una interfaz de túnel sitX en el enrutador principal del sitio UniCauca IPv6

Existen básicamente dos métodos que se pueden utilizar para el establecimiento de un túnel configurado entre UniCauca y la UNAM. El primer método utiliza Scripts de Red para realizar una configuración permanente del túnel sitX y el segundo utiliza comandos de red a través de la consola de Linux para realizar una configuración temporal del túnel sitX.

Nota: Se recomienda utilizar Scripts de Red.

- Configuración de una interfaz de túnel sitX utilizando Scripts de Red

  - Se crea el archivo de configuración de interfaz *ifcfg-<nombre-interfaz>* (ifcfg-**sit1**), en el directorio */etc/sysconfig/network-scripts/* y se agregan las siguientes líneas:

```
DEVICE=sit1
# DEVICE=<nombre-interfaz>, donde <nombre-interfaz> es el nombre de la interfaz a configurar

BOOTPROTO=none
# BOOTPROTO=<protocolo>, donde <protocolo> es uno de los siguientes valores:
# none:  No se utiliza ningún protocolo de tiempo de inicio de S.O
# bootp: Utiliza el protocolo BOOTP
# dhcp:  Utiliza el protocolo DHCP

ONBOOT=yes
# ONBOOT= <indicador>, donde <indicador> es uno de los siguientes valores:
# yes:    El dispositivo de red se activa en el momento de inicio del S.O
# no:     El dispositivo de red no se activa en el momento de inicio del S.O

IPV6INIT=yes
# IPV6INIT=<indicador>, donde <indicador> es uno de los siguientes valores:
# yes:    Habilita la inicialización de IPv6 en ésta interfaz
# no:     Deshabilita la inicialización de IPv6 en ésta interfaz

IPV6TUNNELIPV4=132.248.108.254
# IPV6TUNNELIPV4=<dir-IPv4-equipo-remoto>, donde <dir-IPv4-equipo-remoto> especifica la dirección
#IPv4 del extremo remoto del túnel

IPV6TUNNELIPV4LOCAL=200.30.71.50
# IPV6TUNNELIPV4LOCAL=<dir-IPv4-equipo-local>, donde <dir-IPv4-equipo-local> especifica la
#dirección IPv4 del extremo local del túnel

IPV6ADDR=3FFE:8070:1024::2/64
#IPV6ADDR=<dir-IPv6-equipo-local>/<long-prefijo>, donde <dir-IPv6-equipo-local>/<long-prefijo> es la
#dirección y el prefijo de red IPv6 del extremo local del túnel
```

- Una vez salvado este archivo, se ejecutan desde el directorio */etc/sysconfig/network-scripts/* los siguientes comandos en la consola de Linux:

*ln -s ifup-sit ifup-<nombre-interfaz>; ln -s ifup-sit ifup-sit**1***
*ln -s ifdown-sit ifdown-<nombre-interfaz>; ln -s ifdown-sit ifdown-sit**1***

Lo anterior, permite crear enlaces simbólicos a los Scripts de control de interfaz, encargados de la activación y desactivación de las conexiones de interfaz.

Nota: reiniciar el servicio de red *#service network restart*

- Configuración de una interfaz de túnel sitX utilizando comandos de red

  - Creación de la interfaz de túnel sit**1**

ip tunnel add <nombre-interfaz> mode sit remote <dir-IPv4-equipo-remoto> ...
... local <dir-IPv4-equipo-local> ttl <ttl-default>

ip tunnel add sit1 mode sit remote 132.248.108.254 local 200.30.71.50 ttl 64

  - Activación de la interfaz de túnel sit**1**

ip link set dev <nombre-interfaz> up
*ip link set dev sit**1** up*

  - Asignación de una dirección IPv6 del tipo Global Agregable a la interfaz de túnel sit**1**

ip -6 addr add <dir-IPv6-interfaz-local>/<long-prefijo> dev <interfaz>
*ip -6 addr add 3FFE:8070:1024::2/64 dev sit**1***

3.4.4 **Red piloto UniCauca IPv6: Pruebas de conexión a 6bone**

- Para comprobar que el procedimiento anterior es correcto, desde el enrutador UniCauca IPv6 ejecute los comandos *#ip addr show* e *#ip tunnel show*, para tener una descripción detallada de la configuración de las direcciones IPv4 e IPv6 asignadas a las interfaces de red, así como del túnel IPv6 en IPv4 configurado entre UniCauca y la UNAM. Se deben obtener las siguientes salidas:

```
[root@ipv6 /]# ip addr show

1: lo: <LOOPBACK,UP> mtu 16436 qdisc noqueue
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 brd 127.255.255.255 scope host lo
   inet6 ::1/128 scope host

2: sit0@NONE: <NOARP> mtu 1480 qdisc noop
   link/sit 0.0.0.0 brd 0.0.0.0

3: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 100
   link/ether 00:10:4b:cc:f4:7e brd ff:ff:ff:ff:ff:ff
   inet 200.30.71.50/24 brd 200.30.71.255 scope global eth0
   inet6 fe80::210:4bff:fecc:f47e/64 scope link

4: eth1: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 100
   link/ether 00:b0:d0:c2:d7:76 brd ff:ff:ff:ff:ff:ff
   inet 172.16.41.108/16 brd 172.16.255.255 scope global eth1
   inet6 fe80::2b0:d0ff:fec2:d776/64 scope link
   inet6 3ffe:8070:1024:1::10/64 scope global

10: sit1@NONE: <POINTOPOINT,NOARP,UP> mtu 1480 qdisc noqueue
   link/sit 200.30.71.50 peer 132.248.108.254
   inet6 fe80::c81e:4732/128 scope link
   inet6 3ffe:8070:1024::2/64 scope global
```

```
[root@ipv6 /]# ip tunnel show

sit0: ipv6/ip  remote any  local any  ttl 64  nopmtudisc
sit1: ipv6/ip  remote 132.248.108.254  local 200.30.71.50  ttl 64
```

P.3.1 ¿Interprete la información correspondiente a las interfaces lo, eth0, eth1, sit0 y sit1?

- Ejecute el comando *route –A inet6* para observar la Tabla de Enrutamiento IPv6 del kernel de Linux. Se deberá obtener la siguiente salida:

Kernel IPv6 routing table

Destination	Next Hop	Flags	Metric	Ref	Use	Iface
::1/128	::	U	0	0	0	lo
3ffe:8070:1024::2/128	::	U	0	2	0	lo
3ffe:8070:1024::/64	::	UA	256	2	0	sit1
fe80::ac10:296c/128	::	U	0	0	0	lo
fe80::c81e:4732/128	::	U	0	0	0	lo
fe80::210:4bff:fecc:f47e/128	::	U	0	0	0	lo
fe80::2b0:d0ff:fec2:d776/128	::	U	0	0	0	lo
fe80::/64	::	UA	256	0	0	eth0
fe80::/64	::	UA	256	0	0	eth1
fe80::/64	::	UA	256	0	0	sit1
ff00::/8	::	UA	256	0	0	eth0
ff00::/8	::	UA	256	0	0	eth1
ff00::/8	::	UA	256	0	0	sit1

P.3.2 Analice la Tabla de Enrutamiento IPv6:

a). ¿Que tipos de direcciones IPv6 se presentan?

b). ¿Qué redes se detallan y a través de que interfaces se alcanzan?

c). ¿Describa la función de los parámetros *Next Hop*, *Flags*, *Metric*, *Ref* y *Use*?

- Desde el enrutador UniCauca IPv6 realice un ping6 al extremo *remoto* del túnel, enrutador principal del sitio UNAM IPv6. Para esto ejecute el comando *ping6 3ffe:8070:1024::1*. Deberá obtener la siguiente salida:

```
[root@ipv6 /]# ping6 3ffe:8070:1024::1

PING 3ffe:8070:1024::1(3ffe:8070:1024::1) from 3ffe:8070:1024::2 : 56 data bytes
64 bytes from 3ffe:8070:1024::1: icmp_seq=1 ttl=64 time=2762 ms
64 bytes from 3ffe:8070:1024::1: icmp_seq=2 ttl=64 time=2658 ms
64 bytes from 3ffe:8070:1024::1: icmp_seq=3 ttl=64 time=2423 ms
64 bytes from 3ffe:8070:1024::1: icmp_seq=4 ttl=64 time=2532 ms
64 bytes from 3ffe:8070:1024::1: icmp_seq=5 ttl=64 time=2496 ms
...
```

Mientras se está ejecutando el ping6 al extremo *remoto* del túnel, ejecute el programa analizador de protocolos de red Ethereal[43] sobre la interfaz sit1. Para esto ejecute desde la consola de Linux el comando *ethereal.*

P.3.3 Describa los campos de la cabecera de los protocolos IPv6 e ICMPv6.

[43] Referirse a la sección 3.3.1: Instalación de Red Hat Linux 9 y de las aplicaciones utilizadas en el Laboratorio Internet IPv6. Se recomienda utilizar filtros para los tipos de protocolos IPv6 e ICMPv6. Información detallada acerca del funcionamiento de *ethereal* se encuentra en: http://www.ethereal.com/docs/

P.3.4 ¿Que tipos de mensajes ICMPv6 se observan y cual es su función?

- Desde el enrutador UniCauca IPv6 realice un ping6 a un equipo cualquiera en 6bone. Por ejemplo 3FFE:B00:C18:1::10

P.3.5 ¿Porque este equipo se muestra inalcanzable?

- Agregue una ruta a través del enrutador UNAM IPv6 para alcanzar cualquier destino en 6bone. Para esto ejecute desde el enrutador UniCauca IPv6 el comando */sbin/route -A inet6 add 3FFE::/16 gw 3FFE:8070:1024::1 dev sit1*. Es conveniente revisar la Tabla de Enrutamiento IPv6 con el fin de identificar los cambios realizados. Se debe obtener la siguiente salida:

Kernel IPv6 routing table

Destination	Next Hop	Flags	Metric	Ref	Use	Iface
::1/128	::	U	0	0	0	lo
3ffe:8070:1024::2/128	::	U	0	65	33	lo
3ffe:8070:1024::/64	::	UA	256	4	0	sit1
3ffe::/16	3ffe:8070:1024::1	UG	1	0	0	sit1
fe80::c81e:4732/128	::	U	0	0	0	lo
fe80::210:4bff:fecc:f47e/128	::	U	0	16	13	lo
fe80::2b0:d0ff:fec2:d776/128	::	U	0	8	0	lo
fe80::/64	::	UA	256	0	0	eth0
fe80::/64	::	UA	256	0	0	eth1
fe80::/64	::	UA	256	0	0	sit1
ff00::/8	::	UA	256	0	0	eth0
ff00::/8	::	UA	256	0	0	eth1
ff00::/8	::	UA	256	0	0	sit1
::/0	::	UDA	256	0	0	eth0
::/0	::	UDA	256	0	0	eth1

- Ahora, realice un ping6 a un equipo cualquiera en 6bone para comprobar la alcanzabilidad que se tiene a éste equipo. Por ejemplo 3FFE:B00:C18:1::10

- Como se muestra en la Figura 27, utilizando la interfaz eth1 del enrutador principal del sitio UniCauca IPv6 realice anuncios de enrutamiento sobre el segmento de red que ve ésta interfaz del enrutador. Para esto asigne a la interfaz eth1 la dirección Global Agregable 3FFE:8070:1024:**1**::1/64 y active el demonio de anuncio de rutas *radvd* para anunciar la red 3FFE:8070:1024:**1**::/64 en este segmento. Edite en */etc/radvd.conf* la siguiente información:

```
interface eth1
{
        AdvSendAdvert on;           # Anuncio de enrutamiento (RA) activado
        MinRtrAdvInterval 3;        # Intervalo de anuncios periódicos - 3 a 10 segundos
```

```
        MaxRtrAdvInterval 10;

        prefix 3FFE:8070:1024:1::/64   # Prefijo de red IPv6 a anunciar
        {
                AdvOnLink on;          # Enviar anuncio si se detecta nuevo host en la red
                AdvAutonomous on;
                AdvRouterAddr on;      # Anuncio de ruta IPv6 por defecto
        };
};
```

Nota: Para activar *radvd* ejecute el comando *#service radvd start* en una línea de comandos de la shell.

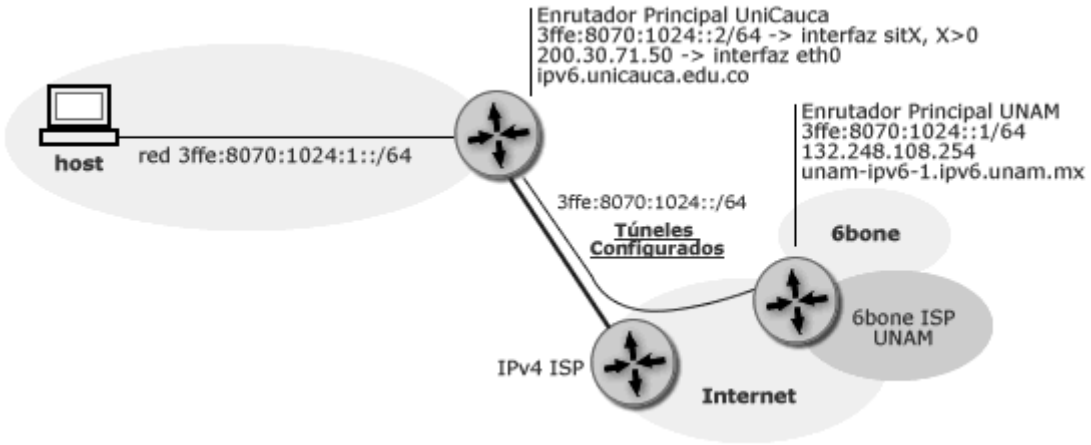


Figura 27 - Escenario de Prueba

Nota: El host que pertenece a la red 3FFE:8070:1024:**1**::/64 fija su dirección Global Agregable por el proceso de *autoconfiguración sin control de estado*, a partir de su Identificador de Interfaz y del prefijo de red anunciado por su enrutador. Si el host es un PC con S.O. MS Windows 2000 con el soporte IPv6 habilitado, ejecute en una consola de Windows el comando *ipv6 if* y asegúrese que este PC haya configurado una dirección IPv6 Unicast Global Agregable. Se debe obtener la siguiente salida:

```
...
Interface X[44] (site 1): Conexión de Área local
 uses Neighbor Discovery
 link-level address: 00-b0-d0-d9-98-7c
    preferred address 3ffe:8070:1024:1:2b0:d0ff:fed9:987c, 2591995s/604795s (addrconf)
    preferred address fe80::2b0:d0ff:fed9:987c, infinite/infinite
    multicast address ff02::1, 1 refs, not reportable
    multicast address ff02::1:ffd9:987c, 2 refs, last reporter
 link MTU 1500 (true link MTU 1500)
 current hop limit 64
 reachable time 22500ms (base 30000ms)
```

[44] X: Depende de la interfaz de red Ethernet que se utiliza.

```
 retransmission interval 1000ms
 DAD transmits 1
...
```

P.3.6 ¿Explique el proceso de creación de la dirección Local de Enlace asignada a la interfaz Ethernet del host?

- Desde el host, realice un ping6 al extremo *local* del túnel, enrutador principal del sitio UniCauca IPv6. Para esto ejecute el comando *ping6 3ffe:8070:1024::**2***. Se deberá obtener la siguiente salida:

```
C:\>ping6 3ffe:8070:1024::2

Pinging 3ffe:8070:1024::2 with 32 bytes of data:

Reply from 3ffe:8070:1024::2: bytes=32 time<1ms
Reply from 3ffe:8070:1024::2: bytes=32 time<1ms
Reply from 3ffe:8070:1024::2: bytes=32 time<1ms
Reply from 3ffe:8070:1024::2: bytes=32 time<1ms
```

P.3.7 ¿Porqué el host alcanza la interfaz sit1 del extremo *local* del túnel?

- Realice un ping6 al extremo *remoto* del túnel, enrutador principal del sitio UNAM IPv6. Para esto ejecute el comando *ping6 3ffe:8070:1024::**1***

P.3.8 ¿Porqué este equipo se muestra inalcanzable?

3.4.5 **Análisis de Resultados**

3.4.6 **Conclusiones**

4 Laboratorio Internet IPv6: Enrutamiento

4.1 Extensión multiprotocolo para BGP-4 con soporte a IPv6 - BGP4+

4.1.1 Motivación

Esta práctica inicialmente explica como configurar el enrutador de frontera del sistema autónomo UniCauca IPv6, utilizando las extensiones IPv6 del multiprotocolo BGP4+, las cuales permiten migrar de una conexión estática a través de un túnel configurado manualmente a una conexión dinámica BGP4+ que garantice la conexión a 6bone. Posteriormente se trabajará en la configuración de escenarios BGP4+ externos (EBGP) e internos (iBGP) que permitan describir las características de BGP4+, así como su funcionamiento, los iguales o vecinos BGP4+, el anuncio de rutas BGP4+, y los problemas de escalabilidad con iBGP - *horizonte dividido BGP4+* -. Los escenarios propuestos facilitan además la interacción permanente con 6bone.

4.1.2 Objetivos

- Configurar BGP4+ en el enrutador de frontera del sistema autónomo UniCauca IPv6 para intercambiar información de enrutamiento con el sistema autónomo UNAM IPv6.

- Configurar un escenario EBGP utilizando además de los sistemas autónomos UniCauca IPv6 y UNAM IPv6, números de sistemas autónomos reservados para uso privado.

- Configurar iBGP en el conjunto de enrutadores que forman el sistema autónomo UniCauca IPv6 para el intercambio de información de enrutamiento interno.

- En todos los casos verificar la conectividad entre sistemas autónomos y hacia 6bone.

4.1.3 Marco teórico

Entender BGP4+, requiere antes de todo entender el concepto de sistemas autónomos.

Sistema autónomo (SA)

La definición clásica de un sistema autónomo es un conjunto de enrutadores bajo una administración técnica única, que utiliza un protocolo de gateway interior y una métrica común para enrutar paquetes en el sistema autónomo y un protocolo de gateway exterior para enrutar paquetes a otros sistemas autónomos.

Los protocolos de enrutamiento pueden clasificarse de acuerdo a si son interiores o exteriores al sistema autónomo. Dos tipos de protocolos de enrutamiento son los siguientes:

- *Interior Gateway Protocol,* Protocolo de gateway interior (IGP). Protocolo de enrutamiento que se usa para intercambiar información de enrutamiento al interior de un sistema autónomo. El *Routing Information Protocol,* Protocolo de información de enrutamiento (RIP), y el *Open Shortest Path First,* primero la ruta libre más corta (OSPF), constituyen ejemplos de IGP. RIPng y OSPFv6, son las versiones de RIP y OSPF para IPv6.

- *Exterior Gateway Protocol,* Protocolo de gateway exterior (EGP). Protocolo de enrutamiento que se usa para conectar sistemas autónomos entre sí. El *Border Gateway Protocol,* Protocolo de gateway fronterizo (BGP), constituye un ejemplo de un EGP. BGP en su versión 4, BGP-4, es la última versión de BGP y BGP4+ es la Extensión multiprotocolo para BGP-4, que da soporte a IPv6. BGP también puede ser usado al interior de un sistema autónomo y esta variación es conocida como BGP interno (iBGP).

4.1.3.1 **Identificador de sistema autónomo (#SA)**

Este identificador de sistema autónomo es un número de 16 bits, entre 1 y 65,536[45]. Un intervalo de números de sistema autónomo, 64,512 a 65,536, queda reservado para el uso privado, de forma similar a las direcciones privadas del protocolo IPv4.

La IANA es la organización encargada de asignar números a los sistemas autónomos. Específicamente, el *American Registry for Internet Numbers,* Registro norteamericano de números de Internet (ARIN) tiene la potestad de asignar números en América, el Caribe y África. El *Reseaux IP Europeennes-Network Information Center,* Redes IP europeas - Centro de información de redes (RIPE-NIC) administra los números de Europa, mientras que *Asia Pacific-NIC* (AP-NIC) administra los números de los sistemas autónomos de la zona Asia-Pacífico.

4.1.3.2 **Terminología y conceptos de BGP4+**

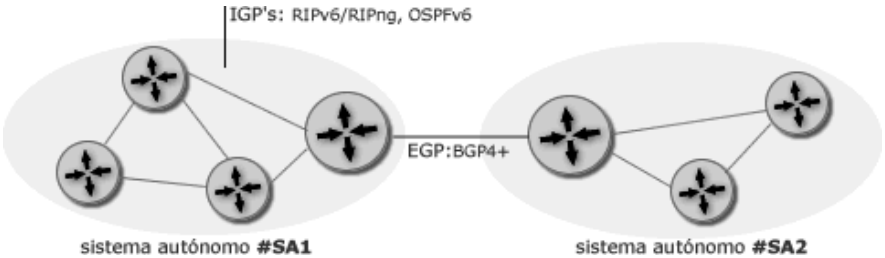


Figura 28 - Los IGP's operan en un sistema autónomo, mientras que los EGP's operan entre SA's

[45] RFC 1930 - Guidelines for Creation, Selection, and Registration of an Autonomous System, Directrices para la creación, selección y registro de un sistema autónomo.

BGP4+ se utiliza entre sistemas autónomos, como se ilustra en la Figura 28. El objetivo principal de BGP4+ consiste en proporcionar un sistema de enrutamiento entre dominios IPv6 que garantice el intercambio sin bucles de información de enrutamiento entre sistemas autónomos.

BGP4+ es un protocolo avanzado de vector distancia, que utiliza el protocolo para control de transmisión (TCP) como protocolo de transporte, lo que garantiza un envío fiable orientado a la conexión, al no tener que implementar ningún mecanismo de retransmisión o de recuperación de errores. BGP4+ utiliza el puerto 179 de TCP, y esta diseñado para soportar los requerimientos de enrutamiento en grandes *internetworks* como 6bone.

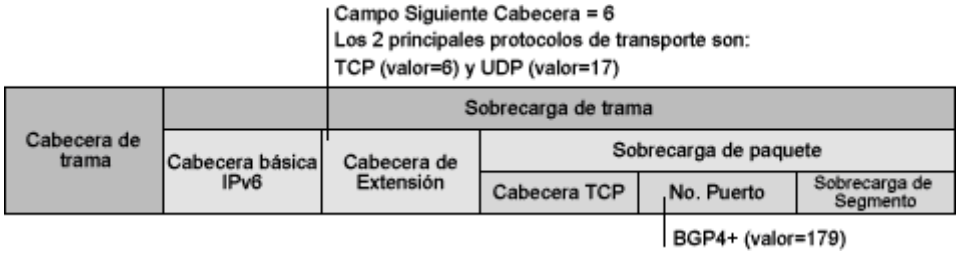


Figura 29 - BGP4+ es transportado dentro de segmentos TCP, que están en el interior de los paquetes IPv6

BGP4+ extiende las capacidades del protocolo de enrutamiento BGP-4, al definir nuevas funcionalidades para dar soporte a la familia de direcciones IPv6, estas son:

- Capacidad para asociar IPv6 con la información de *próximo salto*, que indica la dirección IPv6 del próximo salto que hay que usar para llegar a un destino.

- Capacidad para asociar IPv6 con la información de acceso de nivel de red, *Network Layer Reachability Information* (NLRI), que contiene una lista de prefijos de direcciones IPv6 a las que se puede llegar por una determinada ruta.

4.1.3.3 **Formato de mensajes BGP4+**

Cada mensaje BGP4+ tiene una cabecera de tamaño fijo de 19bytes, aunque puede existir o no una porción de datos después de la cabecera dependiendo del tipo de mensaje BGP4+. La cabecera contiene los siguientes campos:

- Marker (de marcador): Este campo de 16bytes detecta pérdida de sincronización entre un par de enrutadores BGP4+, y se utiliza en algunos casos para autenticar mensajes BGP4+ entrantes.

- Length (de longitud): Este campo de 2bytes indica la longitud total del mensaje BGP4+ en bytes, incluyendo la cabecera[46].

- Type (de tipo): Este campo de 8bits indica el tipo de mensaje.

4.1.3.4 **Tipos de mensajes BGP4+**

BGP4+ define los siguientes tipos de mensajes:

- Open (de inicio)
- Keepalive (de actividad)
- Update (de actualización)
- Notification (de notificación)

Dos enrutadores que comprenden BGP4+ establecen una conexión TCP entre sí e intercambian mensajes para abrir y confirmar los parámetros de conexión. Estos enrutadores se denominan enrutadores iguales o vecinos, y pueden ser internos o externos al sistema autónomo.

Una vez se establece la conexión TCP, el primer mensaje que envía cada enrutador BGP4+ es un mensaje Open. Si el mensaje Open es aceptado, se envía un mensaje Keepalive (en sentido contrario) que confirme la apertura. Cuando ésta se confirma, se establece la conexión BGP4+, y es cuando se pueden intercambiar mensajes de actualización, de actividad y de notificación.

Los enrutadores iguales o vecinos envían mensajes de actualización para intercambiar inicialmente sus tablas de enrutamiento BGP4+ completas, y a partir de este punto, se envían actualizaciones incrementales a medida que cambia la tabla de enrutamiento. Los mensajes de actividad se envían para asegurar que la conexión esté viva entre los enrutadores BGP4+, y los mensajes de notificación son enviados en respuesta a los errores o a las condiciones especiales.

Formato del mensaje Open

Además de la cabecera BGP4+, un mensaje Open incluye la siguiente información:

- Versión: Este campo de 8 bits indica el número de versión del protocolo BGP.

[46] Los mensajes BGP4+ tienen una longitud mínima de 19bytes y una longitud máxima de 4096bytes.

- Mi sistema autónomo: Este campo de 16 bits indica el número de sistema autónomo del emisor.

- Tiempo de espera (hold time): Este campo de 16 bits indica el número *máximo* de segundos que pueden transcurrir entre la recepción de mensajes de actividad o de actualización sucesivos por parte del emisor. Después de recibir un mensaje Open, el enrutador calcula el valor del *temporizador de espera*, utilizando el menor de los tiempos de espera que tiene configurados y el tiempo de espera recibido en el mensaje Open.

- Identificador de enrutador BGP: Este campo de 32 bits en formato de dirección IPv4, indica el identificador BGP del emisor.

- Longitud de parámetros opcionales: Este campo de 1 byte indica la longitud total del campo *parámetros opcionales* en bytes. Si el valor de éste campo es cero, los parámetros opcionales no están presentes.

- Parámetros opcionales: Este campo de longitud variable puede contener una lista de parámetros opcionales.

Formato del mensaje Update

Además de la cabecera BGP4+, un mensaje de actualización incluye principalmente la siguiente información:

- Atributos de ruta: Este campo de longitud variable contiene información sobre el tipo de atributo, la longitud del atributo y el valor del atributo. Estos son: la ruta del sistema autónomo, el próximo salto, el origen, la preferencia local, la comunidad, entre otros.

Formato del mensaje Keepalive

Un mensaje de actividad consiste de una cabecera BGP4+ de longitud 19bytes; y se envía cada 60 segundos por defecto.

Formato del mensaje Notification

Cuando se detecta una condición de error, se envía un mensaje de notificación. La conexión BGP4+ se cierra inmediatamente después de que éste mensaje es enviado. Los mensajes de notificación incluyen un código de error, un subcódigo de error y datos relacionados con el error.

4.1.3.5 **Atributos de ruta BGP**

Una vez BGP recibe actualizaciones sobre distintos destinos desde distintos sistemas autónomos, el protocolo decide qué ruta va a elegir para alcanzar un destino específico. BGP elige una sola ruta para llegar a un destino específico, y el proceso de decisión de ruta se basa en los atributos de ruta que BGP implementa.

Los atributos de ruta incluyen información sobre la métrica BGP, y se dividen en las siguientes categorias:

- Atributos bien conocidos, obligatorios
- Atributos bien conocidos, discrecionales
- Atributos opcionales, transitivos
- Atributos opcionales, no transitivos

Atributos bien conocidos

Son aquellos que todas las implementaciones BGP deben reconocer. Estos atributos se propagan a los vecinos BGP.

Un atributo obligatorio bien conocido debe aparecer en la descripción de la ruta. Un atributo discrecional bien conocido no necesita aparecer en una descripción de ruta.

Atributos opcionales

Un atributo opcional no necesita ser soportado por todas las implementaciones BGP; podría tratarse de un atributo privado. Si se soporta, puede ser propagado a los vecinos BGP.

Un atributo transitivo opcional que no está implementado en un enrutador debe ser pasado a otros enrutadores BGP. En este caso, el atributo esta marcado como parcial. Un atributo no transitivo opcional debe ser eliminado por un enrutador que no haya implementado el atributo.

4.1.3.6 **Atributos de ruta BGP4+ definidos**

Los atributos más significativos que define BGP4+ son los siguientes:

- Atributos bien conocidos, obligatorios:
  - Origen
  - Ruta de sistema autónomo
  - Próximo salto

- Atributos bien conocidos, discrecionales:
  - Preferencia local

- Atributos opcionales, transitivos:
  - Comunidad

- Atributos opcionales, no transitivos[47]: Dos nuevos atributos que define BGP4+, permiten tener compatibilidad con las versiones anteriores de BGP. Estos son:

  - Información de acceso de nivel de red alcanzable
  - Información de acceso de nivel de red inalcanzable

Atributo de Origen: Define el origen de la información de ruta. Este atributo puede ser uno de estos tres valores:

- IGP: La ruta es interna al sistema autónomo que la origina. Un origen IGP se indica con una "**i**" en la tabla BGP.

- EGP: La ruta se conoce a través del Protocolo de gateway exterior. Un origen EGP se indica con una "**e**" en la tabla BGP.

- Incompleto: El origen de la ruta es desconocido o se conoce a través de otros medios. Un origen Incompleto se indica con un signo "**?**" en la tabla BGP.

Atributo de ruta de sistema autónomo: Este atributo es en realidad la lista de números de sistema autónomo que una ruta ha atravesado para llegar a un destino. Siempre que una actualización de ruta atraviesa un sistema autónomo, el número de sistema autónomo se antepone a esa actualización.

El atributo de ruta de sistema autónomo lo utilizan los enrutadores BGP4+ para asegurar un entorno sin bucles. Si un enrutador BGP4+ recibe una ruta en la que su propio sistema autónomo es parte del atributo de ruta de sistema autónomo, no aceptará la ruta.

Atributo de próximo salto: Indica la dirección IPv6 de próximo salto que hay que usar para llegar a un destino.

Para BGP4+ actuando como un EGP (EBGP), el próximo salto es la dirección IPv6 del vecino que envió la actualización. En la Figura 30, El enrutador **A** publicará la red

[47] Si un enrutador BGP no soporta la Extensión multiprotocolo para BGP-4, esta en capacidad de ignorar la información transportada por estos dos atributos.

3FFE:8070:1024::/48 en el enrutador **B**, mientras que el enrutador **B** publicará la red 3FFE:8070:1000::/48 en el enrutador **A**. Entonces, el enrutador **A** utilizará 3FFE:8070:1024::**1**/64 como atributo de próximo salto para alcanzar la red 3FFE:8070:1000::/48, mientras que el enrutador **B** utilizará 3FFE:8070:1024::**2**/64 como atributo de próximo salto para alcanzar la red 3FFE:8070:1024::/48.

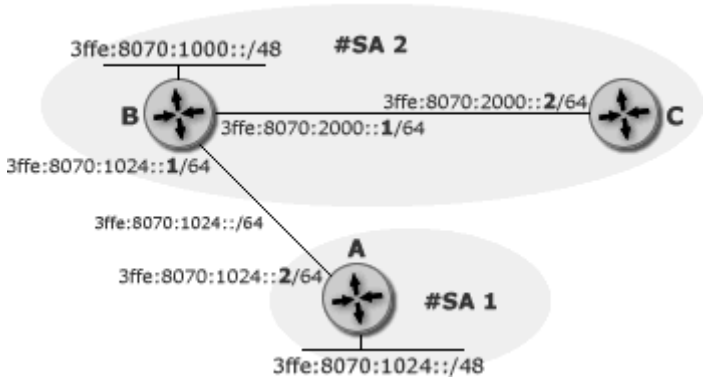


Figura 30 - Atributo de próximo salto BGP4+

Para BGP4+ actuando como un iBGP, el protocolo indica que el próximo salto publicado por EGP (EBGP) debe ser llevado a iBGP. Debido a esto, el enrutador **B** publicará la red 3FFE:8070:1024::/48 en su vecino iBGP (enrutador **C**), con un próximo salto de 3FFE:8070:1024::**2**/64 (la dirección del enrutador **A**). Por lo tanto, el enrutador **C** sabrá que el próximo salto para alcanzar la red 3FFE:8070:1024::/48 es 3FFE:8070:1024::**2**/64 y no 3FFE:8070:2000::**1**/64, como se podría esperar. El enrutador **C** debe saber llegar a la red 3FFE:8070:1024::/64 (que conecta a los enrutadores **A** y **B**), a través de un IGP o de una ruta estática.

Atributo de preferencia local: Proporciona una indicación a los enrutadores del sistema autónomo acerca de qué ruta es la preferida para salir del sistema autónomo. Es preferible una ruta que tenga una preferencia local más alta. La preferencia local es un atributo que se configura en un enrutador y que se intercambia solamente entre los enrutadores del mismo sistema autónomo.

Atributo comunidad: Las comunidades BGP4+ constituyen una forma de filtrar rutas entrantes o salientes, ya que permiten a los enrutadores etiquetar rutas con un indicador (la **comunidad**) y facilitan a otros enrutadores tomar decisiones con base en esta etiqueta. Las comunidades son utilizadas por las rutas que comparten algunas propiedades comunes, por lo que también comparten normas comunes; de esta forma, los enrutadores actúan sobre la comunidad, en vez de sobre rutas individuales.

Atributo de información de acceso de nivel de red alcanzable: Se utiliza para transportar la lista de direcciones IPv6 a los que se puede llegar por una ruta, junto con la información de próximo salto que es utilizada para alcanzar determinados destinos.

Atributo de información de acceso de nivel de red inalcanzable: Se utiliza para transportar la lista de destinos IPv6 inalcanzables.

4.1.4 **Desarrollo técnico**

4.1.4.1 **Implementación del multiprotocolo BGP4+ en Red Hat Linux**

Para implementar BGP4+ en Red Hat Linux se utilizará *zebra*[48], el cual es un software de enrutamiento avanzado distribuido bajo Licencia Pública GNU (GPL) que administra diferentes protocolos de enrutamiento basados en TCP/IP.

Nota: Desde una consola de Linux ejecute el comando *rpm –qa | grep zebra*, para verificar la instalación del paquete *zebra* en el equipo de cómputo. Si *zebra* no está instalado, los CD-ROM's de RH Linux 9 facilitan este paquete en formato RPM, para esto ejecute el comando *rpm –i zebra-0.93b-1.i386.rpm.*

Arquitectura del sistema zebra - BGP4+

Zebra es un software modular, donde cada protocolo de enrutamiento es un servicio independiente y es gestionado por un servicio principal llamado *zebra*, el kernel gestor del enrutamiento[49]. La arquitectura de sistema *zebra* para BGP4+ se muestra a continuación:

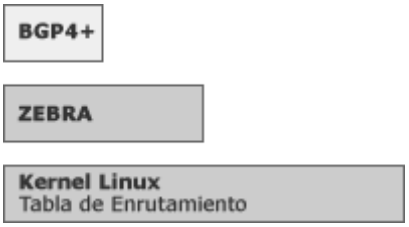


Figura 31 - Arquitectura del Sistema zebra

Trabajando con zebra - BGP4+

En el directorio */etc/zebra/* se deben crear los archivos de configuración *zebra.conf* y *bgpd.conf*, necesarios para la configuración de estos 2 servicios.

[48] Otras implementaciones: GateD*****, MRTd*****, Kame BGPd [***** están bajo desarrollo]
[49] La modularidad que ofrece *zebra* permite que cada servicio tenga su propio archivo de configuración e Interfaz de terminal.

- El archivo *zebra.conf* deberá tener las siguientes entradas:

```
hostname zebra-ipv6
password ipv6
enable password ipv6
log stdout
```

- El archivo *bgpd.conf* deberá tener las siguientes entradas:

```
hostname bgpd-ipv6
password ipv6
enable password ipv6
log stdout
```

- Inicie los servicios *zebra* y *bgpd,* y entre al archivo de configuración de *bgpd* mediante una conexión *telnet* al puerto *2605* que corresponde a *bgpd.* Puede acceder al servicio *bgpd* desde el propio equipo (localhost) o bien desde equipos remotos.

[root@ipv6 /]# service zebra start
iniciando zebra: *[OK]*
[root@ipv6 /]# service bgpd start
iniciando bgpd *[OK]*

[root@ipv6 /]# telnet localhost ***2605***
Trying 127.0.0.1...
Connected to localhost.

Hello, this is zebra (version 0.93b).
Copyright 1996-2002 Kunihiro Ishiguro.

User Access Verification
Password:

- Ingrese el *Password* que por defecto será “ipv6” y estará en el *modo EXEC* (comandos BGP en modo terminal) del servicio BGP.

bgpd-ipv6>

Nota: Cada una de las tareas presentadas a continuación se identifica como requerida u opcional para la configuración básica de BGP4+.

Tareas:

- Configuración de un proceso de enrutamiento BGP4+ y un identificador de enrutador BGP *(requerida)*
- Configuración de iguales o vecinos BGP4+ *(requerida)*
- Anuncio de rutas a través de BGP4+ *(requerida)*
- Configuración de confederación de SA's *(opcional)*
- Verificación de la configuración y operación de BGP4+ *(opcional)*

Configuración de un proceso de enrutamiento BGP4+ y un identificador de enrutador BGP

Prerrequisitos[50]: Antes de configurar el enrutador para que corra BGP4+, confirme que la interfaz del enrutador que anunciará las redes del SA UniCauca a 6bone disponga de una dirección IPv6 Unicast Global Agregable, y habilite el *reenvió* (forwarding) de tráfico global IPv6 en el enrutador[51].

BGP usa un ID de enrutador para identificar enrutadores iguales o vecinos que comprenden BGP4+, éste es un valor de 32 bits que se representa por la sintaxis de una dirección IPv4 y que se puede configurar manualmente de la siguiente manera:

- Habilite los privilegios del *modo EXEC*
  bgpd-ipv6*>enable*
  Password: ipv6
  bgpd-ipv6*#*

- Acceda al modo de configuración general
  bgpd-ipv6# *configure terminal*
  bgpd-ipv6 *(config)#*

- Acceda al *modo de configuración de enrutador.* De esta manera se activa el proceso de enrutamiento BGP con el #SA especificado.

  bgpd-ipv6 *(config)# router bgp* ***#SA***
  Ejemplo: ***bgpd-ipv6*** *(config)# router bgp* ***64515***

- Deshabilite la familia de direcciones Unicast IPv4 para el proceso de enrutamiento BGP especificado en el paso anterior.

[50] Para una mayor comprensión, referirse a las Sección 3.3.3: Configuración IPv6 de red en RH Linux
[51] Verifique el parámetro IPV6FORWARDING=yes en el archivo */etc/sysconfig/network.*

bgpd-ipv6 *(config-router)# no bgp default ipv4-unicast*

- Configure un ID como identificador para el enrutador local que corre BGP.

  bgpd-ipv6 *(config-router)# bgp router-id* ***dir-IPv4***
  Ejemplos: ***bgpd-ipv6*** *(config-router)# bgp router-id* ***200.30.71.50***
  bgpd-ipv6 *(config-router)# bgp router-id* ***172.16.41.X***[52]

- Guarde los cambios y salga del modo de configuración de enrutador.

  bgpd-ipv6 *(config-router)# write memory*
  bgpd-ipv6 *(config-router)# end*
  bgpd-ipv6*#disable*
  bgpd-ipv6*>*

Configuración de iguales o vecinos BGP4+

Los enrutadores iguales o vecinos que se definen utilizando el comando ***neighbor remote-as*** en el *modo de configuración de enrutador*, permiten el intercambio de prefijos de direcciones IPv4 e IPv6. Para el caso específico de IPv6, los iguales o vecinos pueden ser activados usando el comando ***neighbor activate*** en el *modo de configuración de familia de direcciones.*

- Habilite los privilegios del *modo EXEC*

  bgpd-ipv6*>enable*
  Password: ipv6
  bgpd-ipv6*#*

- Acceda al modo de configuración general

  bgpd-ipv6# *configure terminal*
  bgpd-ipv6 *(config)#*

- Acceda al *modo de configuración de enrutador.* De esta manera se activa el proceso de enrutamiento BGP con el #SA especificado.

  bgpd-ipv6 *(config)# router bgp* ***#SA***
  Ejemplo: ***bgpd-ipv6*** *(config)# router bgp* ***64515***

[52] 100 ≤ **X** ≤ 140: rango de direcciones IPv4 asignado al Laboratorio de Telecomunicaciones de la FIET.

- Agregue la dirección IPv6 Unicast Global Agregable de la interfaz IPv6 exterior del enrutador igual o vecino, y especifique el #SA remoto.

  bgpd-ipv6 *(config-router)# neighbor* ***dir-IPv6*** *remote-as* ***#SA***
  Ejemplos: ***bgpd-ipv6*** *(config-router)# neighbor* ***3FFE:8070:1024::1*** *remote-as* ***278***
  bgpd-ipv6 *(config-router)# neighbor* ***3FFE:8070:1024:1::10*** *remote-as* ***64515***

- Especifique la interfaz IPv6 local que le permite alcanzar a su vecino BGP4+.

  bgpd-ipv6 *(config-router)# neighbor* ***dir-IPv6*** *interface* ***nombre-interfaz***
  Ejemplos: ***bgpd-ipv6*** *(config-router)# neighbor* ***3FFE:8070:1024::1*** *interface* ***sit1***
  bgpd-ipv6 *(config-router)# neighbor* ***3FFE:8070:1024:1::10*** *interface* ***eth1***

- Especifique la familia de direcciones IPv6 y entre al *modo de configuración de familia de direcciones.*

  bgpd-ipv6 *(config-router)# address-family ipv6 unicast*
  bgpd-ipv6 *(config-router-af)#*

  Nota: *unicast,* especifica la familia de direcciones IPv6 Unicast. Si la palabra clave *unicast,* no se especifíca, el *modo de configuración de familia de direcciones* la establece por defecto.

- Habilite el intercambio de prefijos de direcciones IPv6, entre el enrutador local y el vecino.

  bgpd-ipv6 *(config-router-af)# neighbor* ***dir-IPv6*** *activate*
  Ejemplos: ***bgpd-ipv6*** *(config-router-af)# neighbor* ***3FFE:8070:1024::1*** *activate*
  bgpd-ipv6 *(config-router-af)# neighbor* ***3FFE:8070:1024:1::10*** *activate*

- Guarde los cambios y salga del modo de configuración de familia de direcciones.

  bgpd-ipv6*(config-router-af)# write memory*
  bgpd-ipv6*(config-router-af)# end*
  bgpd-ipv6*#disable*
  bgpd-ipv6*>*

Anuncio de rutas a través de BGP4+

Las redes que son definidas usando el comando ***network*** en el *modo de configuración de enrutador*, se inyectan por defecto en una base de datos IPv4 BGP Unicast. Para inyectar una red en una base de datos IPv6 BGP Unicast, se debe especificar la red usando el comando ***network*** en el *modo de configuración de familia de direcciones.*

- Habilite los privilegios del *modo EXEC*

  bgpd-ipv6*>enable*
  Password: ipv6
  bgpd-ipv6*#*

- Acceda al modo de configuración general

  bgpd-ipv6# *configure terminal*
  bgpd-ipv6*(config)#*

- Acceda al *modo de configuración de enrutador.* De esta manera se activa el proceso de enrutamiento BGP con el #SA especificado.

  bgpd-ipv6 *(config)# router bgp* ***#SA***
  Ejemplo: ***bgpd-ipv6*** *(config)# router bgp* ***64515***

- Especifique la familia de direcciones IPv6 y entre al *modo de configuración de familia de direcciones.*

  bgpd-ipv6 *(config-router)# address-family ipv6 unicast*
  bgpd-ipv6 *(config-router-af)#*

- Anuncie el prefijo de red IPv6 en la base de datos IPv6 BGP Unicast (las rutas se pueden localizar en la tabla de enrutamiento IPv6 Unicast).

  bgpd-ipv6 *(config-router-af)# network* ***dir-IPv6/long-prefijo***
  Ejemplos: ***bgpd-ipv6*** *(config-router-af)# network* ***3FFE:8070:1024::/48***
  bgpd-ipv6 *(config-router-af)# network* ***3FFE:8070:1024:1::/64***

- Guarde los cambios y salga del modo de configuración de familia de direcciones.

  bgpd-ipv6 *(config-router-af)# write memory*

bgpd-ipv6 *(config-router-af)# end*
bgpd-ipv6*#disable*
bgpd-ipv6*>*

Configuración de Confederación de SA's

Si en un sistema autónomo se tiene iBGP como IGP, los anuncios conocidos por un enrutador vía iBGP no se anunciarán vía iBGP a otro igual o vecino BGP4+. En estas circunstancias el comando ***bgp confederation*** en el *modo configuración de enrutador* permite establecer una confederación de sistemas autónomos para BGP que facilita el intercambio de información de enrutamiento entre los distintos vecinos BGP4+.

- Habilite los privilegios del *modo EXEC*

  bgpd-ipv6*>enable*
  Password: ipv6
  bgpd-ipv6*#*

- Acceda al modo de configuración general

  bgpd-ipv6# *configure terminal*
  bgpd-ipv6*(config)#*

- Acceda al *modo de configuración de enrutador.* De esta manera se activa el proceso de enrutamiento BGP con el #SA especificado.

  bgpd-ipv6 *(config)# router bgp* ***#SA***
  Ejemplo: ***bgpd-ipv6*** *(config)# router bgp* ***64520***

- Especifique los parámetros de la confederación de SA's. Estos son: *identifier* #***SA*** que indica el número de sistema autónomo que la confederación anunciará al exterior, y los *peers* #***sub-SA1*** #***sub-SA2*** #***sub-SAn*** *...* que indican los subsistemas autónomos vecinos que pertenecen a la confederación.

  bgpd-ipv6 *(config-router)# bgp confederation identifier* ***#SA***
  Ejemplo: ***bgpd-ipv6*** *(config)# bgp confederation identifier* ***64515***

  bgpd-ipv6 *(config-router)# bgp confederation peers #****sub-SA1*** *#****sub-SA2*** *#****sub-SAn****...*
  Ejemplo: ***bgpd-ipv6*** *(config)# bgp confederation peers* ***64525 64530***

- Guarde los cambios y salga del modo de configuración de enrutador.

bgpd-ipv6 *(config-router)# write memory*
bgpd-ipv6 *(config-router)# end*
bgpd-ipv6*#disable*
bgpd-ipv6*>*

Verificación de la configuración y operación de BGP4+

La verificación de la configuración de BGP4+ puede llevarse a cabo por medio de los siguientes comandos *EXEC show*:

- Despliegue la configuración actual BGP4+ salvada en el archivo */etc/zebra/bgpd.conf*
bgpd-ipv6*#*
bgpd-ipv6*# show running-config*

- Despliegue las entradas de la tabla de enrutamiento BGP4+
bgpd-ipv6*>*
bgpd-ipv6*> show bgp ipv6*

- Despliegue el estado de todas las conexiones BGP4+
bgpd-ipv6*>*
bgpd-ipv6*> show bgp ipv6 summary*

- Despliegue las características de todas las conexiones BGP4+ establecidas con los vecinos BGP4+
bgpd-ipv6*>*
bgpd-ipv6*> show bgp ipv6 neighbors*

- Despliegue las entradas de la tabla de enrutamiento BGP4+ para una red IPv6 específica
bgpd-ipv6*>*
bgpd-ipv6*> show bgp ipv6 red-IPv6/Long Prefijo IPv6*

- Despliegue de rutas anunciadas desde el enrutador IPv6 local a su vecino BGP4+
bgpd-ipv6*>*
bgpd-ipv6*> show bgp ipv6 neighbor* ***dir-IPv6*** *advertised-routes*

- Despliegue de rutas aprendidas desde el enrutador vecino BGP4+
bgpd-ipv6*>*
bgpd-ipv6*> show bgp ipv6 neighbor* ***dir-IPv6*** *routes*

4.1.5 **Recursos**

- 3 PC's con S.O Red Hat Linux 9 (enrutadores IPv6)
- 1 PC's con S.O Red Hat Linux 9 (host IPv6)
- 1 PC's con S.O MS Windows 2000 (host IPv6)
- Dirección IPv4 local oficial: 200.30.71.50
- #SA UniCauca: 64515
- #SA UNAM: 278
- Proveedor de servicios IPv6 (6bone ISP): UNAM
- Espacio de direcciones IPv6 (*bloque* pNLA /48): 3FFE:8070:1024::/48

4.1.6 **Desarrollo de la Práctica**

Esta práctica se divide en 3 fases:

- Fase 1: Habilitación de BGP4+ externo (EBGP)
- Fase 2: Habilitación de BGP4+ externo en un escenario con múltiples SA's
- Fase 3: Habilitación de BGP4+ interno (iBGP)

Fase 1: Habilitación de BGP4+ externo (EBGP)

En esta fase se migra del actual túnel IPv6 en IPv4 configurado manualmente entre la Universidad del Cauca y la UNAM, a un túnel dinámico que utiliza las extensiones IPv6 del multiprotocolo BGP4+. La implementación de BGP4+ como un EBGP entre sistemas autónomos garantiza la conexión del sitio UniCauca IPv6 a 6bone.

La Figura 32, describe la topología que se debe implementar:

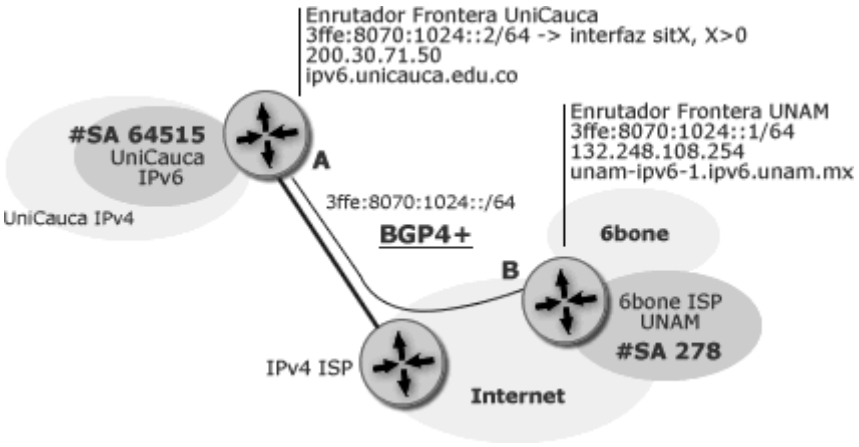


Figura 32 -Topología de la Fase 1

La configuración de las interfaces de red Ethernet en el enrutador de frontera del SA UniCauca IPv6 es la siguiente:

Enrutador Linux	**A**
Interfaz sit**X, X>0**	3FFE:8070:1024::2/64
Interfaz Ethernet [eth0]	200.30.71.50
Interfaz Ethernet [eth1]	3FFE:8070:1024:**1**::1/64

Tabla 5 - Configuración dual de direcciones IPv4/IPv6 en el enrutador de frontera del SA UniCauca IPv6

Notas:

- Referirse a la Sección 3.3.2: Soporte IPv6 en RH Linux
- Referirse a la Sección 3.3.3: Configuración IPv6 de red en RH Linux
- Para configurar interfaces IPv4, se utiliza la Herramienta de administración de redes de RH Linux. Para esto, se ejecuta el comando *#redhat-config-network* en una línea de comandos de la shell.

4.1.7 **Procedimiento Fase 1**

Este procedimiento se basa en cada una de las tareas presentadas anteriormente para la configuración básica de BGP4+ en un equipo enrutador IPv6 utilizando *zebra*.

- En el enrutador **A**, configure el proceso de enrutamiento BGP4+ utilizando el #SA 64515.

- En el enrutador **A**, configure iguales o vecinos BGP4+, para establecer al enrutador **B** como igual o vecino BGP4+.

- En el enrutador **A**, anuncie rutas a través de BGP4+ para publicar el espacio de direcciones IPv6 3FFE:8070:1024::/48 al SA UNAM IPv6.

- En el enrutador **A** ejecute el comando ***bgpd-ipv6****# show running-config* para desplegar la información actual BGP4+ salvada en el archivo */etc/zebra/bgpd.conf* y así verificar su correcta configuración.

- En el enrutador **A**, introduzca los comandos ***bgpd-ipv6****> show bgp ipv6 summary* y ***bgpd-ipv6****> show bgp ipv6 neighbors*.

P.4.1.1 ¿Cuál es el estado BGP del enrutador **B**?

P.4.1.2 ¿Cuánto tiempo lleva establecida la relación de vecindad con el enrutador **B**?

- En el enrutador **A**, introduzca el comando ***bgpd-ipv6***> *show bgp ipv6 neighbors 3FFE:8070:1024::1*

P.4.1.3 Identifique los atributos de próximo salto: *Nexthop, Nexthop global y Nexthop local*, que observa el enrutador **B**

P.4.1.4 ¿Qué tipo de conexión establecen los enrutadores **A** y **B**?

- En el enrutador **A**, introduzca el comando ***bgpd-ipv6***> *show bgp ipv6.*

P.4.1.5 ¿Cuál es el número total de prefijos de red que aparece en la tabla de enrutamiento BGP4+?

P.4.1.6 ¿Cúal es la interpretación que se puede dar a la lista de números de sistema autónomo que aparecen en el Atributo de ruta de sistema autónomo (Path)?

P.4.1.7 ¿Qué indica un origen IGP que se muestra con una "**i**" en la tabla de enrutamiento BGP4+?

- Como se muestra en la Figura 33, utilizando la interfaz eth1 del enrutador principal del sitio UniCauca IPv6 realice anuncios de enrutamiento sobre el segmento de red que ve ésta interfaz del enrutador. Para esto asigne a la interfaz eth1 la dirección Global Agregable 3FFE:8070:1024:**1**::10/64 y active el demonio de anuncio de rutas *radvd* para anunciar la red 3FFE:8070:1024:**1**::/64 en este segmento. Edite en */etc/radvd.conf* la siguiente información:

```
interface eth1
{
        AdvSendAdvert on;              # Anuncio de enrutamiento (RA) activado
        MinRtrAdvInterval 3;           # Intervalo de anuncios periódicos - 3 a 10 segundos
        MaxRtrAdvInterval 10;

        prefix 3FFE:8070:1024:1::/64   # Prefijo de red IPv6 a anunciar
        {
                AdvOnLink on;          # Enviar anuncio si se detecta nuevo host en la red
                AdvAutonomous on;
                AdvRouterAddr on;      # Anuncio de ruta IPv6 por defecto
        };
};
```

Nota: Para activar *radvd* ejecute el comando *#service radvd start* en una línea de comandos de la shell.

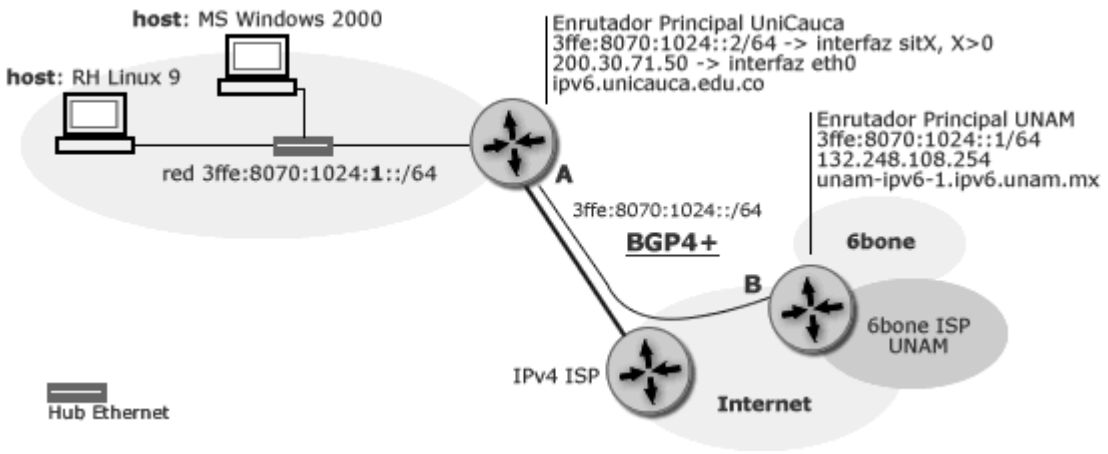


Figura 33 - Topología de la Fase 1: Escenario de Prueba

Nota: Los hosts que pertenece a la red 3FFE:8070:1024:**1**::/64 fija su dirección Global Agregable por el proceso de *autoconfiguración sin control de estado*, a partir de su Identificador de Interfaz y del prefijo de red anunciado por el enrutador **A**.

- Desde el host MS Windows 2000 realice el siguiente proceso:

  - C:\>ping6 3ffe:8070:1024:1::1
  - C:\>ping6 3ffe:8070:1024::2
  - C:\>ping6 3ffe:8070:1024::1

P.4.1.8 Observe y analice los tiempos de respuesta en cada caso, ¿Qué conclusiones obtiene?

- Desde el host RH Linux 9 realice el siguiente proceso:

  - [host@ipv6 /]# ping6 3ffe:8070:1024:1::1
  - [host@ipv6 /]# ping6 3ffe:8070:1024::2
  - [host@ipv6 /]# ping6 3ffe:8070:1024::1

P.4.1.9 Observe y analice los tiempos de respuesta en cada caso, ¿Qué conclusiones obtiene?

P.4.1.10 Realice una comparación del desempeño de las 2 plataformas operativas anteriores en el contexto de IPv6.

- Desde cada uno de los hosts realice un ping6 a un equipo cualquiera en 6bone. Por ejemplo 3FFE:B00:C18:1::10

P.4.1.11 Es alcanzable este equipo, ¿Por qué si o por qué no?

- Desde cada uno de los hosts realice un trazado de ruta a un equipo cualquiera en 6bone. Por ejemplo 3FFE:B00:C18:1::10. Para esto utilice el comando *traceroute6* desde Red Hat Linux y *tracert6* desde MS Windows 2000.

P.4.1.12 ¿Qué muestran los comandos de trazado de ruta IPv6 y que concluye de esto?

- Utilice el analizador de protocolos de red Ethereal para identificar los tipos de mensajes BGP4+ que envía y recibe el enrutador **A**. Para esto ejecute desde la consola de Linux el comando # *ethereal* sobre la interfaz sitX, X>0.

  Nota: Reinicie el servicio *bgpd* con el comando # *service bgpd restart*.

P.4.1.13 Describa el formato del mensaje OPEN

P.4.1.14 Describa el formato del mensaje KEEPALIVE

P.4.1.15 Describa el formato del mensaje UPDATE

- Desconecte el cable de red de la interfaz eth0, para observar el mensaje que se genera cuando se detecta una condición de error.

P.4.1.16 Describa el formato del mensaje NOTIFICATION

P.4.1.17 Describa el proceso de encapsulación de BGP4+ para su transporte sobre TCP/IPv6.

Fase 2: Habilitación de BGP4+ externo en un escenario con múltiples SA's

En esta fase se estudiará el funcionamiento de BGP4+ como EBGP en un escenario donde se tienen enrutadores IPv6 que corren BGP4+ y que pertenecen a diferentes sistemas autónomos.

La Figura 34, describe la topología que se debe implementar:

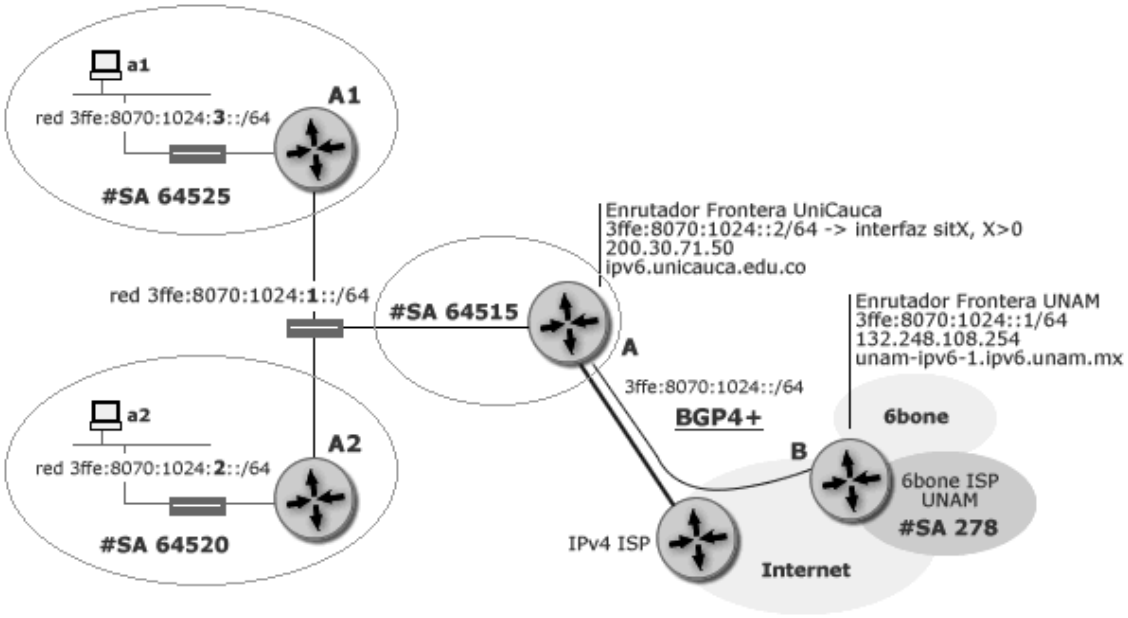


Figura 34 - Topología Fase 2

La configuración de las interfaces de los enrutadores es la siguiente:

Enrutador Linux	**A**	**A1**	**A2**
Interfaz sit**X, X>0**	3FFE:8070:1024::**2**/64	-	-
Interfaz Ethernet [eth0]	200.30.71.50	3FFE:8070:1024:1::**10**/64	3FFE:8070:1024:1::**20**/64
Interfaz Ethernet [eth1]	3FFE:8070:1024:1::**1**/64	3FFE:8070:1024:3::**10**/64	3FFE:8070:1024:2::**20**/64

Tabla 6 - Configuración de direcciones IPv6 en los enrutadores

Nota: Los hosts **a1** y **a2** que pertenecen a las redes 3FFE:8070:1024:**3**::/64 y 3FFE:8070:1024:**2**::/64 respectivamente, fijan su dirección IPv6 Unicast Global Agregable por el proceso de *autoconfiguración sin control de estado*, a partir de su Identificador de Interfaz y del prefijo de red anunciado por su enrutador. El host **a1** es un equipo con S.O MS Windows 2000 y el host **a2** es un equipo con S.O RH Linux 9.

4.1.8 **Procedimiento Fase 2**

Este procedimiento se basa en cada una de las tareas presentadas anteriormente para la configuración básica de BGP4+ en un equipo enrutador IPv6 utilizando *zebra*.

Nota: El enrutador **A** ya dispone de una configuración BGP4+ previa, realizada en la Fase 1. Ahora configure lo siguiente:

- En el enrutador **A**, configure iguales o vecinos BGP4+, para establecer a los enrutadores **A1** y **A2** como iguales o vecinos BGP4+.

- En el enrutador **A1**, configure el proceso de enrutamiento BGP4+ utilizando el #SA 64525.

- En el enrutador **A1**, configure iguales o vecinos BGP4+, para establecer a los enrutadores **A** y **A2** como iguales o vecinos BGP4+.

- En el enrutador **A1**, anuncie rutas a través de BGP4+ para publicar la red IPv6 3FFE:8070:1024:**3**::/64 a los SA's vecinos.

- En el enrutador **A2**, configure el proceso de enrutamiento BGP4+ utilizando el #SA 64520.

- En el enrutador **A2**, configure iguales o vecinos BGP4+, para establecer a los enrutadores **A** y **A1** como iguales o vecinos BGP4+.

- En el enrutador **A2**, anuncie rutas a través de BGP4+ para publicar la red IPv6 3FFE:8070:1024:**2**::/64 a los SA's vecinos.

- En los enrutadores **A**, **A1** y **A2** ejecute el comando ***bgpd-ipv6****# show running-config* para desplegar la información actual BGP4+ salvada en el archivo */etc/zebra/bgpd.conf* y así verificar su correcta configuración.

- En los enrutadores **A**, **A1** y **A2**, introduzca el comando ***bgpd-ipv6***> *show bgp ipv6* para observar la tabla de enrutamiento BGP4+ de cada uno de los enrutadores.

P.4.1.18 ¿Cuál es el número total de prefijos de red que aparece en la tabla de enrutamiento BGP4+ de cada enrutador?

P.4.1.19 ¿Comparten los 3 enrutadores la misma tabla de enrutamiento?. Justifique su respuesta.

P.4.1.20 Analice la información que transporta el Atributo de ruta de sistema autónomo (Path) en cada enrutador. ¿Existen diferencias en la lista de números de sistema autónomo que una ruta debe atravesar para llegar a un destino?. Justifique su respuesta.

- Ejecute en cada uno de los enrutadores los comandos: ***bgpd-ipv6***> *show bgp ipv6 neighbors* ***dir-IPv6*** *advertised-routes*, ***bgpd-ipv6***> *show bgp ipv6 neighbors* ***dir-IPv6*** *routes*, donde ***dir-IPv6*** representa la dirección IPv6 del enrutador igual o vecino BGP4+ de acuerdo a la siguiente relación de vecindad:

	A	A1	A2	B
A	-	3FFE:8070:1024:1::**10**	3FFE:8070:1024:1::**20**	3FFE:8070:1024::**1**
A1	3FFE:8070:1024:1::**1**	-		
A2		3FFE:8070:1024:1::**10**	-	-

Tabla 7 - Relación de vecindad entre los enrutadores A, A1 y A2

P.4.1.21 ¿Cuáles son las redes anunciadas por cada enrutador?

P.4.1.22 ¿Cuántos prefijos de red aprendió cada enrutador?

- Desde los hosts **a1** y **a2** realice un ping6 a un equipo cualquiera en 6bone. Por ejemplo 3FFE:B00:C18:1::10

P.4.1.23 Es alcanzable este equipo, ¿Por qué si o por qué no?

- Desde los hosts **a1** y **a2** realice un trazado de ruta a un equipo cualquiera en 6bone. Por ejemplo 3FFE:B00:C18:1::10. Para esto utilice el comando *traceroute6* desde Red Hat Linux y *tracert6* desde MS Windows 2000.

P.4.1.24 ¿Qué muestran los comandos de trazado de ruta IPv6 y que concluye de esto?

  - Desconecte el cable de red de la interfaz eth1 en el enrutador **A**, para dar de baja el enlace que existe entre el enrutador **A** y los enrutadores **A1** y **A2**. En **A1** y **A2** ejecute repetidas veces el comando: ***bgpd-ipv6***> *show bgp ipv6 neighbors 3FFE:8070:1024:1::1 advertised-routes* y ***bgpd-ipv6***> *show bgp ipv6 neighbors 3FFE:8070:1024:1::1 routes.*

P.4.1.25 ¿Qué concluye del tiempo de convergencia de BGP4+ al producirse un cambio en la topología de la red?

Fase 3: Habilitación de BGP4+ interno (iBGP)

En esta fase se estudiará el funcionamiento de BGP4+ como iBGP al interior del SA UniCauca IPv6 para identificar los problemas de escalabilidad asociados con BGP4+ interno. Seguidamente se utilizará la técnica de Confederación de SA's para integrar esta fase con BGP4+ externo (EBGP), permitiendo así, que el escenario propuesto interactué permanentemente con 6bone.

La Figura 35, describe la topología que se debe implementar:

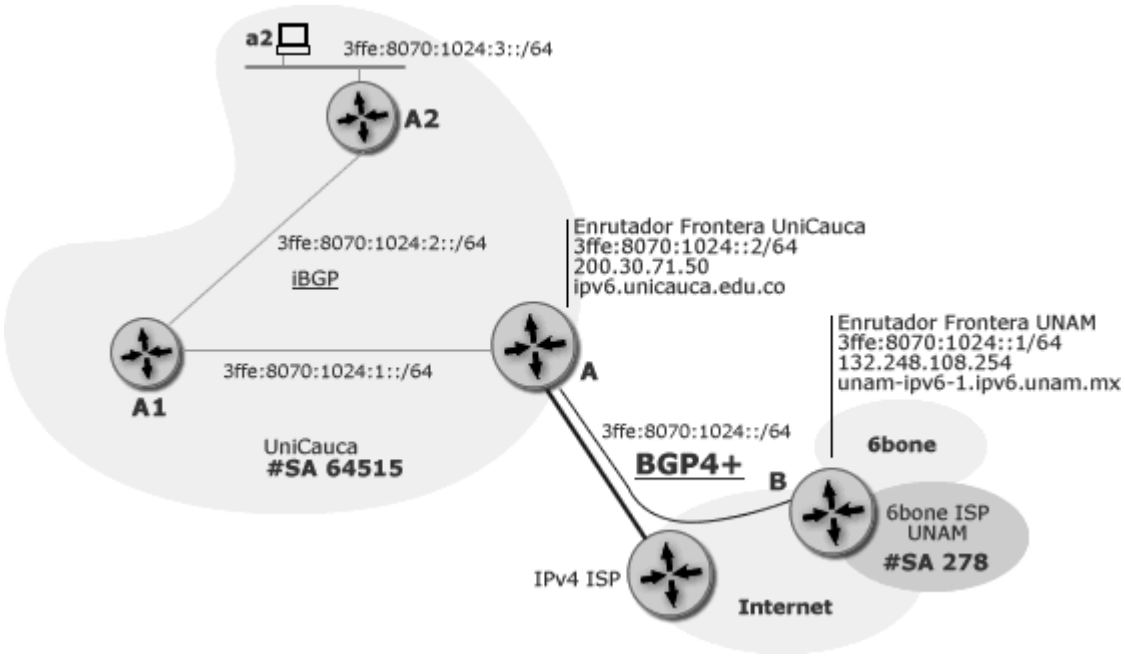


Figura 35 - Topología de la Fase 3

La configuración de las interfaces de los enrutadores es la siguiente:

Enrutador Linux	**A**	**A1**	**A2**
Interfaz sit**X, X>0**	3FFE:8070:1024::**2**/64	-	-
Interfaz Ethernet [eth0]	200.30.71.50	3FFE:8070:1024:1::**10**/64	3FFE:8070:1024:2::**20**/64
Interfaz Ethernet [eth1]	3FFE:8070:1024:1::**1**/64	3FFE:8070:1024:2::**10**/64	3FFE:8070:1024:3::**20**/64

Tabla 8 - Configuración de direcciones IPv6 en los enrutadores

Nota: El host **a2** que pertenece a la red 3FFE:8070:1024:**3**::/64, fija su dirección IPv6 Unicast Global Agregable por el proceso de *autoconfiguración sin control de estado*, a partir de su Identificador de Interfaz y del prefijo de red anunciado por su enrutador.

Particularidades de BGP4+ interno - iBGP

Para que dentro de un sistema autónomo que se habla iBGP todos los enrutadores conozcan las redes anunciadas vía iBGP, éstos enrutadores deben estar completamente interconectados - *malla cerrada* -, es decir, se deben establecer sesiones BGP4+ de todos con todos. Esto se debe a que los anuncios conocidos por un enrutador vía iBGP no se anuncian vía iBGP a otro igual o vecino BGP4+ como mecanismo para evitar bucles de información de enrutamiento.

Para solucionar esto se han desarrollado 2 técnicas:

- Reflectores de ruta
- Confederación de SA's

Reflectores de ruta: Éstos modifican la regla del horizonte dividido[53] BGP4+ permitiendo configurar el enrutador como reflector de ruta para propagar las rutas conocidas por iBGP a otros iguales iBGP, como se muestra en la Figura 36. Con esto se ahorra el número de sesiones TCP/BGP que se debe mantener y se reduce el tráfico de enrutamiento BGP4+.

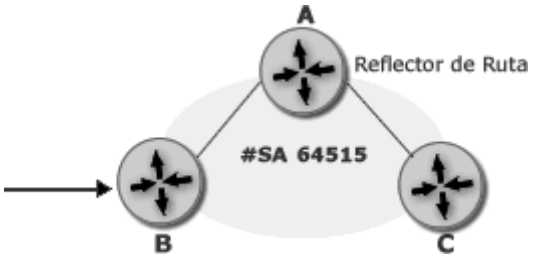


Figura 36 - Cuando el enrutador **A** es un Reflector de Ruta, puede propagar las rutas conocidas del enrutador **B** al enrutador **C**

Confederación de Sistemas Autónomos: Esta técnica consiste en subdividir el sistema autónomo en subsistemas autónomos manejables, establecer BGP4+ al interior de cada nuevo subsistema que se comportará como si fuese un SA independiente e integrar éstos subsistemas en una confederación que se anunciará al exterior como un solo SA, como se ilustra en la Figura 37.

[53] Técnica de enrutamiento en la que se impide que la información sobre las rutas salga de la interfaz del enrutador a través del cual se recibió esa información. Las actualizaciones de horizonte dividido resultan útiles a la hora de impedir bucles de enrutamiento.

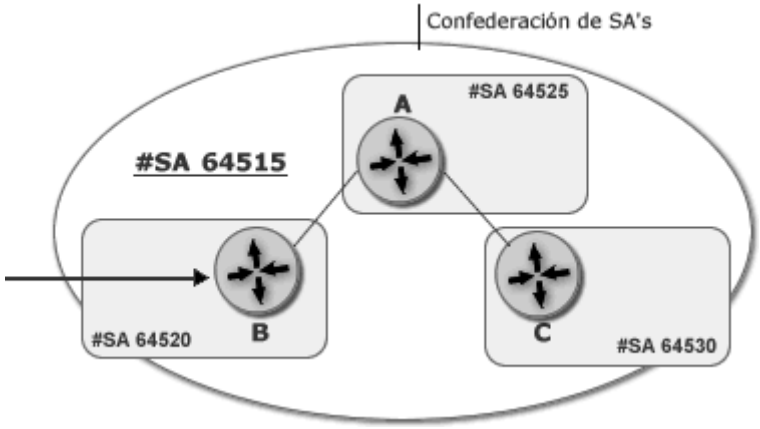


Figura 37 - Confederación de SA's

Nota: Esta fase se realizará utilizando la técnica de Confederaciones de SA, ya que esta técnica exige un mayor grado de dificultad en su implementación.

4.1.9 **Procedimiento Fase 3**

Este procedimiento se basa en cada una de las tareas presentadas anteriormente para la configuración básica de BGP4+ en un equipo enrutador IPv6 utilizando *zebra*.

Paso 1: Topología en malla abierta - Topología propuesta inicialmente para la Fase 3

Nota: El enrutador **A** ya dispone de una configuración BGP4+ previa, realizada en la Fase 1. Ahora configure lo siguiente:

- En el enrutador **A**, configure iguales o vecinos BGP4+, para establecer al enrutador **A1** como igual o vecino BGP4+.

- En el enrutador **A1**, configure el proceso de enrutamiento BGP4+ utilizando el #SA 64515.

- En el enrutador **A1**, configure iguales o vecinos BGP4+, para establecer a los enrutadores **A** y **A2** como iguales o vecinos BGP4+.

- En el enrutador **A1**, anuncie rutas a través de BGP4+ para publicar las redes IPv6 3FFE:8070:1024:**1**::/64 y 3FFE:8070:1024:**2**::/64 a sus vecinos BGP4+.

- En el enrutador **A2**, configure el proceso de enrutamiento BGP4+ utilizando el #SA 64515.

- En el enrutador **A2**, configure iguales o vecinos BGP4+, para establecer al enrutador **A1** como igual o vecino BGP4+.

- En el enrutador **A2**, anuncie rutas a través de BGP4+ para publicar la red IPv6 3FFE:8070:1024:**2**::/64 a su vecino BGP4+.

- En los enrutadores **A**, **A1** y **A2** ejecute el comando ***bgpd-ipv6****# show running-config* para desplegar la información actual BGP4+ salvada en el archivo */etc/zebra/bgpd.conf* y así verificar su correcta configuración.

- En los enrutadores **A**, **A1** y **A2**, introduzca el comando ***bgpd-ipv6***> *show bgp ipv6* para observar la tabla de enrutamiento BGP4+ de cada uno de los enrutadores.

P.4.1.26 ¿Cuál es el número total de prefijos de red que aparece en la tabla de enrutamiento BGP4+ de cada enrutador?

P.4.1.27 ¿Comparten los 3 enrutadores la misma tabla de enrutamiento?. Justifique su respuesta.

- Ejecute en cada uno de los enrutadores los comandos: ***bgpd-ipv6***> *show bgp ipv6 neighbors* ***dir-IPv6*** *advertised-routes*, ***bgpd-ipv6***> *show bgp ipv6 neighbors* ***dir-IPv6*** *routes*, donde ***dir-IPv6*** representa la dirección IPv6 del enrutador igual o vecino BGP4+ de acuerdo a la siguiente relación de vecindad:

	A	**A1**	**A2**	**B**
A	-	3FFE:8070:1024:1::**10**	-	3FFE:8070:1024::**1**
A1	3FFE:8070:1024:1::**1**	-	3FFE:8070:1024:1::**20**	-
A2	-	3FFE:8070:1024:1::**10**	-	

Tabla 9 - Relación de vecindad entre los enrutadores A, A1, A2 y B

P.4.1.28 ¿Cuáles son las redes anunciadas por cada enrutador?

P.4.1.29 ¿Cuántos prefijos de red aprendió cada enrutador?

P.4.1.30 Establezca la diferencia entre las rutas anunciadas y las rutas aprendidas por cada uno de los enrutadores.

P.4.1.31 ¿Qué concluye de los problemas de escalabilidad asociados con iBGP en la topología de red propuesta?

Paso 2: Confederación de Sistemas Autónomos - Nueva Topología Fase 3

- La Figura 38, describe la nueva topología que se debe implementar:

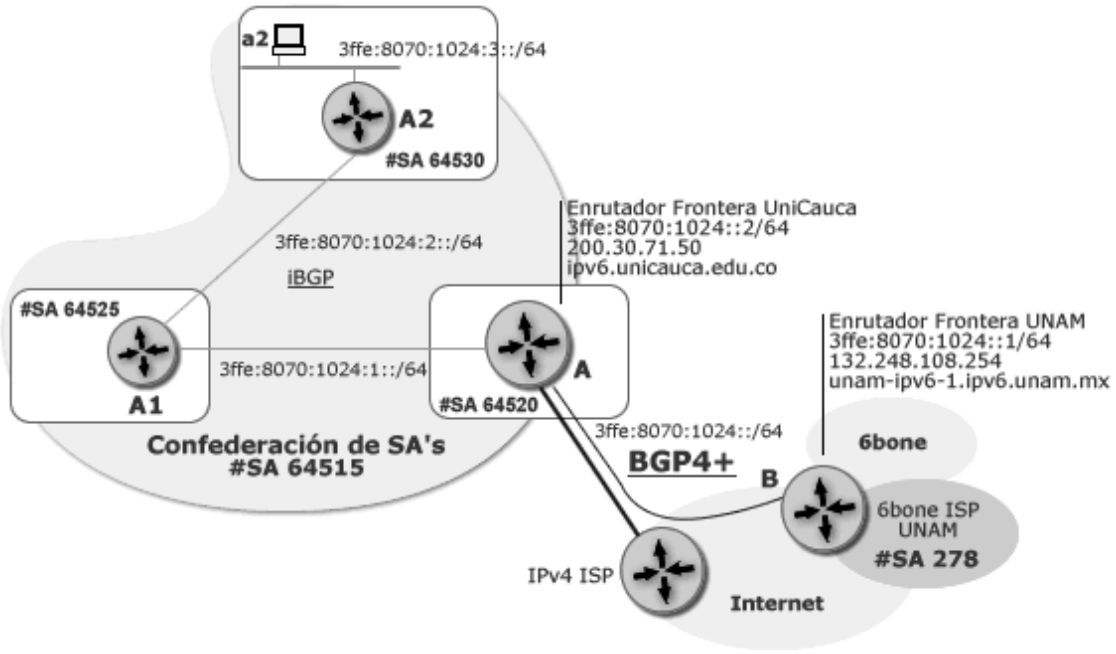


Figura 38 - Nueva Topología de la Fase 3

Nota: La configuración de las interfaces de red Ethernet en cada uno de los enrutadores es igual a la presentada inicialmente para esta fase.

- Es aconsejable iniciar con las entradas originales configuradas en el archivo */etc/zebra/bgpd.conf* para cada uno de los enrutadores de trabajo. Entonces el archivo *bgpd.conf* deberá tener las siguientes entradas:

```
hostname bgpd-ipv6
password ipv6
enable password ipv6
log stdout
```

- Para trabajar con los nuevos parámetros BGP4+ configurados, se reinicia el servicio *bgpd* ejecutando el comando *#service bgpd restart*

- En el enrutador **A**, configure el proceso de enrutamiento BGP4+ utilizando el #SA 64520.

- En el enrutador **A**, configure una confederación de SA, utilizando como identificador de confederación el #SA 64515 y como vecinos BGP los subsistemas autónomos 64525 y 64530.

- En el enrutador **A**, configure iguales o vecinos BGP4+, para establecer a los enrutadores **A1** y **B** como iguales o vecinos BGP4+.

- En el enrutador **A**, anuncie rutas a través de BGP4+ para publicar el espacio de direcciones IPv6 3FFE:8070:1024::/48 a sus vecinos BGP4+.

- En el enrutador **A1**, configure el proceso de enrutamiento BGP4+ utilizando el #SA 64525.

- En el enrutador **A1**, configure una confederación de SA, utilizando como identificador de confederación el #SA 64515 y como vecinos BGP los subsistemas autónomos 64520 y 64530.

- En el enrutador **A1**, configure iguales o vecinos BGP4+, para establecer a los enrutadores **A** y **A2** como iguales o vecinos BGP4+.

- En el enrutador **A1**, anuncie rutas a través de BGP4+ para publicar las redes IPv6 3FFE:8070:1024:**1**::/64 y 3FFE:8070:1024:**2**::/64 a sus vecinos BGP4+.

- En el enrutador **A2**, configure el proceso de enrutamiento BGP4+ utilizando el #SA 64530.

- En el enrutador **A2**, configure una confederación de SA, utilizando como identificador de confederación el #SA 64515 y como vecinos BGP los subsistemas autónomos 64525 y 64520.

- En el enrutador **A2**, configure iguales o vecinos BGP4+, para establecer al enrutador **A1** como igual o vecino BGP4+.

- En el enrutador **A2**, anuncie rutas a través de BGP4+ para publicar la red IPv6 3FFE:8070:1024:**3**::/64 a su vecino BGP4+.

- En los enrutadores **A**, **A1** y **A2** ejecute el comando ***bgpd-ipv6****# show running-config* para desplegar la información actual BGP4+ salvada en el archivo */etc/zebra/bgpd.conf* y así verificar su correcta configuración.

- En los enrutadores **A**, **A1** y **A2**, introduzca el comando ***bgpd-ipv6***> *show bgp ipv6* para observar la tabla de enrutamiento BGP4+ de cada uno de los enrutadores.

P.4.1.32 ¿Cuál es el número total de prefijos de red que aparece en la tabla de enrutamiento BGP4+ de cada enrutador?

P.4.1.33 ¿Comparten los 3 enrutadores la misma tabla de enrutamiento?. Justifique su respuesta.

P.4.1.34 En los enrutadores **A1** y **A2** analice la información que transporta el Atributo de ruta de sistema autónomo (Path) y que precede al #SA 278. ¿Existen diferencias en la lista de números de sistema autónomo que una ruta debe atravesar para llegar a un destino?. Justifique su respuesta.

- Ejecute en cada uno de los enrutadores los comandos: ***bgpd-ipv6***> *show bgp ipv6 neighbors* ***dir-IPv6*** *advertised-routes*, ***bgpd-ipv6***> *show bgp ipv6 neighbors* ***dir-IPv6*** *routes*, donde ***dir-IPv6*** representa la dirección IPv6 del enrutador igual o vecino BGP4+ de acuerdo a la siguiente relación de vecindad:

	A	**A1**	**A2**	**B**
A	-	3FFE:8070:1024:1::**10**	-	3FFE:8070:1024::**1**
A1	3FFE:8070:1024:1::**1**	-	3FFE:8070:1024:1::**20**	-
A2	-	3FFE:8070:1024:1::**10**	-	

Tabla 10 - Relación de vecindad entre los enrutadores A, A1, A2 y B

P.4.1.35 ¿Cuáles son las redes anunciadas por cada enrutador?

P.4.1.36 ¿Cuántos prefijos de red aprendió cada enrutador?

4.1.10 **Análisis de Resultados**

4.1.11 **Conclusiones**

4.2 Protocolo RIP con soporte a IPv6 - RIPng

4.2.1 Motivación

Esta práctica explica como explorar las capacidades del protocolo RIPng, sobre diversos escenarios de enrutamiento propuestos para la red piloto UniCauca IPv6. Para esto se trabajará en 2 fases: Intercambio de mensajes en RIPng y Coexistencia entre RIPng como IGP y BGP4+ como EGP en el SA UniCauca IPv6.

4.2.2 Objetivos

- Habilitar y configurar RIPng como un IGP en el conjunto de enrutadores que forman la red piloto UniCauca IPv6 para intercambiar información de enrutamiento al interior del sistema autónomo UniCauca IPv6.

- Redistribuir BGP4+ en RIPng para estudiar las capacidades que tienen los enrutadores fronterizos de conectar distintos sistemas autónomos con el fin de intercambiar y publicar información de enrutamiento recibida de un sistema autónomo a otro sistema autónomo.

- Verificar la conectividad en el sistema autónomo UniCauca IPv6 y hacia 6bone.

4.2.3 Marco teórico

RIPng[54] es la adaptación del *Routing Information Protocol*, Protocolo de información de enrutamiento (RIP) que implementa extensiones para IPv6, es un *Interior Gateway Protocol*, Protocolo de gateway interior (IGP) que se usa habitualmente en redes IPv6 pequeñas y medianas. RIPng es consecuencia directa de los protocolos de enrutamiento por vector distancia, define una métrica de conteo de saltos (*number of hops*) equivalente al número de enrutadores vecinos que deben atravesarse para alcanzar un destino, reside en la capa de aplicación y posee un número de puerto UDP[55] de 521. Soporta la arquitectura de direccionamiento IPv6 y la utilización del grupo de direcciones Multicast FF02::9 como dirección destino para los mensajes de actualización RIPng.

[54] RIP para IPv6, versión 3 del protocolo de vector distancia RIP. La especificación de RIP para IPv6 es publicada como un Request For Comments (RFC) 2080.
[55] User Datagram Protocol, Protocolo de datagrama de usuario.

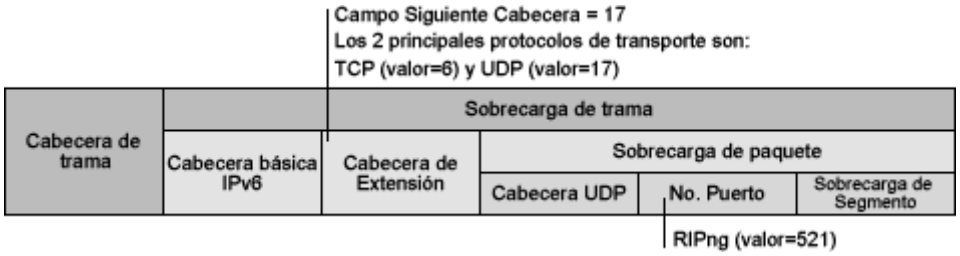


Figura 39 - RIPng es transportado dentro de segmentos UDP, que están en el interior de los paquetes IPv6

RIPng divide los equipos de una comunidad IPv6 en activos y pasivos, los equipos activos anuncian sus rutas a los otros; los equipos pasivos listan y actualizan sus rutas con base en estos anuncios, pero no anuncian.

Un enrutador que corre RIPng en modo activo publica periódicamente a sus vecinos un mensaje de actualización que contiene información tomada de su base de datos de enrutamiento IPv6. Cada mensaje consiste de un prefijo de dirección de red IPv6 seguido de su longitud y de un entero entre 1 y 15 *inclusive* que representa la distancia hacia esta red; si existe una ruta con más de esta distancia, RIPng la considera inalcanzable.

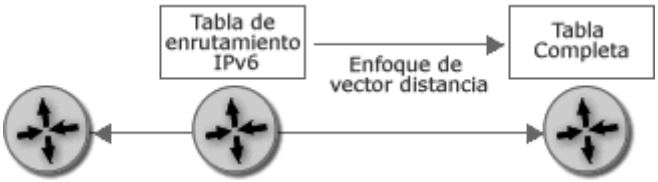


Figura 40 - RIPng envía toda la tabla de enrutamiento IPv6 a sus vecinos

Cuando un equipo RIPng activo como pasivo recibe una tabla completa de un enrutador vecino, puede verificar todas las rutas conocidas y luego realizar cambios en la tabla IPv6 local con base en la información actualizada recibida. Esta tabla de enrutamiento define una entrada por cada red IPv6 destino que es alcanzable a través del proceso de enrutamiento RIPng, y cada entrada contiene al menos la siguiente información:

- Prefijo de dirección de red IPv6 destino / Longitud del prefijo.
- Métrica RIPng, la cual representa la suma total de los costos asociados con las redes que se deben atravesar para alcanzar el destino.
- Dirección IPv6 del enrutador de próximo salto.
- Una bandera, que indica la información de rutas que han cambiado recientemente.
- Temporizadores asociados con las rutas establecidas.

En la topología RIPng de la Figura 41, el enrutador **A1** anunciará un mensaje en la red 3FFE:8070:1024:**2**::/64, dando a entender que puede alcanzar la red 3FFE:8070:1024:**1**::/64 a

la distancia 1[56]. El enrutador **A2** recibirá el anuncio e instalará una ruta hacia la red 3FFE:8070:1024:**1**::/64 a través de **A1** a la distancia 2. Después, el enrutador **A2** anunciará un mensaje en la red 3FFE:8070:1024:**3**::/64 y continuando el proceso el enrutador **A3** instalará finalmente una ruta hacia la red 3FFE:8070:1024:**1**::/64 a través de **A2** a la distancia 3 en su tabla de enrutamiento IPv6.

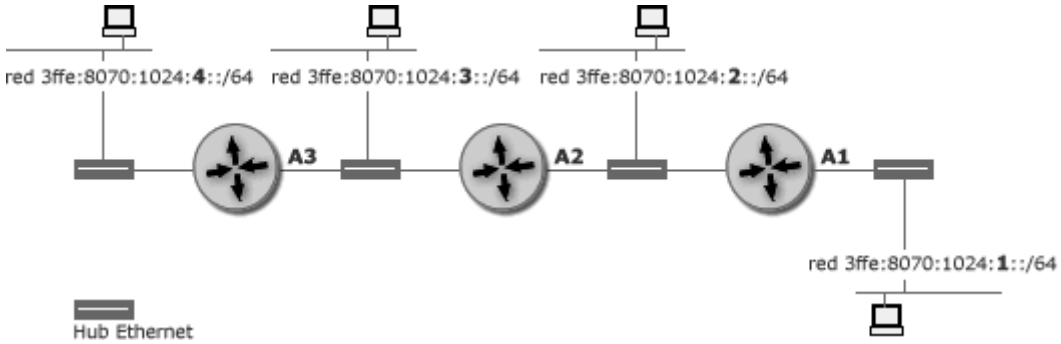


Figura 41 - Funcionamiento de RIPng

Cuando se sincronizan todas las tablas de enrutamiento IPv6 y cada una de ellas contiene una ruta utilizable a cada red destino, significa que la red *converge*. La convergencia es la actividad que está asociada con que las tablas de enrutamiento se sincronicen cuando se produce un cambio en la topología de red.

La secuencia de eventos de la convergencia RIPng cuando el enrutador **A3** de la Figura 42 detecta el fallo en la red 3FFE:8070:1024::**1**/64 es la siguiente:

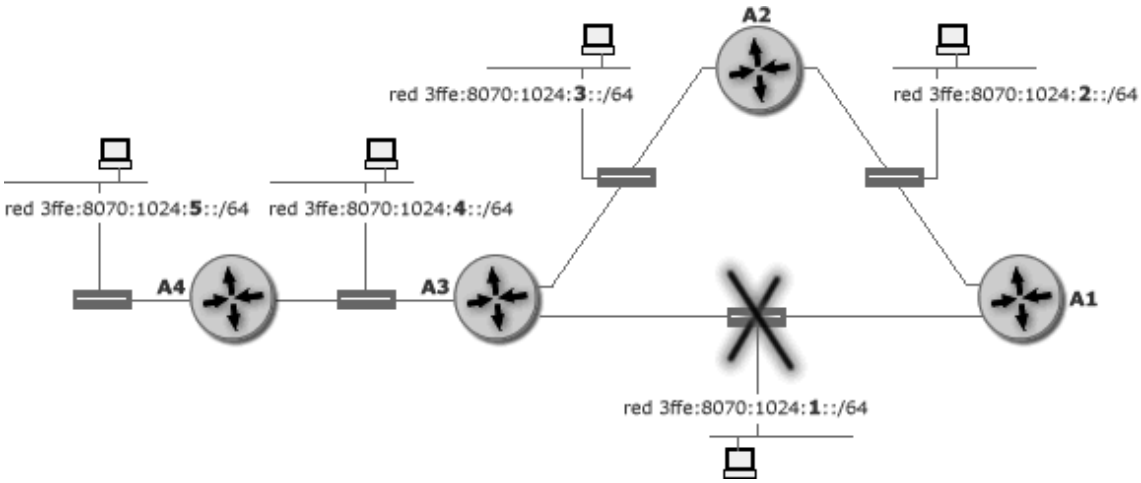


Figura 42 - Convergencia de RIPng

El enrutador **A3** detecta el fallo del enlace Ethernet entre los enrutadores **A1** y **A3**. El enrutador **A3** envía una actualización *flash* (una actualización que se envía cuando se produce

[56] RIP permite a los administradores de una red configurar manualmente el valor de la métrica de red de acuerdo a condiciones estimadas.

un cambio, en vez de en el intervalo periódico normal), incluyendo una ruta envenenada (una ruta que tiene una métrica inalcanzable, en el caso de RIPng, una cuenta de salto 16), a los enrutadores **A2** y **A4**. El enrutador **A3** elimina la entrada del enlace no válido de su tabla de enrutamiento IPv6 y también elimina todas las rutas que estén asociadas con este enlace.

El enrutador **A3** envía una consulta a sus vecinos, buscando una ruta alternativa a la red 3FFE:8070:1024::**1**/64.

El enrutador **A4** responde con una ruta envenenada a la red 3FFE:8070:1024::**1**/64 , y el enrutador **A2** responde con una ruta válida a la red 3FFE:8070:1024::**1**/64. El enrutador **A3** instala inmediatamente la ruta desde el enrutador **A2**. El enrutador **A3** no entra en espera, puesto que la entrada inválida ya se había eliminado de su tabla de enrutamiento IPv6.

El enrutador **A4** entra en espera por la ruta inválida. Cuando el enrutador **A3** difunde su mensaje de actualización, el enrutador **A4** ignorará la ruta debido a que está en espera (durante la espera, se ignoran las rutas que tengan la misma métrica o una métrica mayor que la que tenía un enrutador originalmente para una red). El enrutador **A4** sigue enviando una ruta envenenada al enrutador **A3** en sus mensajes de actualización.

Cuando el enrutador **A4** sale de la espera, la nueva ruta publicada por el enrutador **A3** hará que se actualicen las entradas de su tabla de enrutamiento IPv6.

4.2.4 **Desarrollo técnico**

4.2.4.1 **Implementación de RIPng en Red Hat Linux**

Para implementar RIPng en Red Hat Linux se utilizará *zebra*, el cual es un software de enrutamiento avanzado distribuido bajo Licencia Pública GNU (GPL) que administra diferentes protocolos de enrutamiento basados en TCP/IP.

Nota: Desde una consola de Linux ejecute el comando *rpm –qa* | *grep zebra*, para verificar la instalación del paquete zebra en el equipo de cómputo. Si zebra no está instalado, los CD-ROM's de RH Linux 9 facilitan este paquete en formato RPM, para esto ejecute el comando *rpm –i zebra-0.93b-1.i386.rpm*.

Arquitectura del sistema zebra - RIPng

Zebra es un software modular, donde cada protocolo de enrutamiento es un servicio independiente y es gestionado por un servicio principal llamado *zebra*, el kernel gestor del enrutamiento[57]. La arquitectura de sistema *zebra* para RIPng se muestra a continuación:

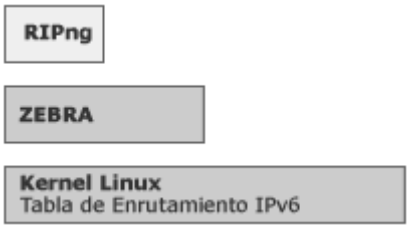


Figura 43 - Arquitectura del Sistema Zebra

Trabajando con zebra - RIPng

En el directorio */etc/zebra/* se deben crear los archivos de configuración *zebra.conf* y *ripngd.conf*, necesarios para la configuración de estos 2 servicios.

- El archivo *zebra.conf* deberá tener las siguientes entradas:

```
hostname zebra-ipv6
password ipv6
enable password ipv6
log stdout
```

- El archivo *ripngd.conf* deberá tener las siguientes entradas:

```
hostname ripngd-ipv6
password ipv6
enable password ipv6
log stdout
```

- Inicie los servicios *zebra* y *ripngd,* y acceda al archivo de configuración *ripngd.conf* mediante una conexión *telnet* al puerto *2603* que corresponde a *ripngd.* Puede acceder al servicio *ripngd* desde el propio equipo (localhost) o bien desde equipos remotos.

[root@ipv6 /]# service zebra start
iniciando zebra: *[OK]*

[57] La modularidad que ofrece *zebra* permite que cada servicio tenga su propio archivo de configuración e Interfaz de terminal.

[root@ipv6 /]# service ripngd start
iniciando ripngd [OK]

*[root@ipv6 /]# telnet localhost **2603***
Trying 127.0.0.1...
Connected to localhost.

Hello, this is zebra (version 0.93b).
Copyright 1996-2002 Kunihiro Ishiguro.

User Access Verification
Password:

- Ingrese el Password que por defecto será "ripngd" y estará en el modo *EXEC* (comandos generales) del servicio RIPng.

  ripngd-ipv6>

Nota: Cada una de las tareas descritas a continuación se identifica como requerida u opcional en la implementación de RIPng.

Tareas:

- Habilitando RIPng - modo activo *(requerida)*
- Habilitando RIPng - modo pasivo *(opcional)*
- Temporizar RIPng *(opcional)*
- Redistribuir BGP4+ en RIPng *(opcional)*
- Verificación de la configuración y operación de RIPng *(opcional)*

Habilitando RIPng - modo activo

Prerrequisitos[58]: Antes de configurar los enrutadores para que corran RIPng en *modo activo*, confirme que las interfaces que anunciarán las redes dentro del SA UniCauca dispongan de una dirección IPv6 Unicast Global Agregable, y habilite el *reenvió* (forwarding) de tráfico global IPv6 en los enrutadores[59].

[58] Para una mayor comprensión, referirse a la Sección 3.3.3: Configuración IPv6 de red en RH Linux.
[59] Verifique el parámetro IPV6FORWARDING=yes en el archivo */etc/sysconfig/network.*

- Habilite los privilegios del *modo EXEC*

  ripngd-ipv6*>enable*
  Password: ipv6
  ripngd-ipv6*#*

- Acceda al modo de configuración general

  ripngd-ipv6# *configure terminal*
  ripngd-ipv6 *(config)#*

- Acceda al *modo de configuración de enrutador.* De esta manera activa el proceso de enrutamiento RIPng.

  ripngd-ipv6*(config)# router ripng*
  ripngd-ipv6*(config-router)#*

- Habilite el envío / recepción de información de actualización RIPng.

  ripngd-ipv6*(config-router)# network NETWORK / IFNAME*

  Nota:

  NETWORK, RIPng únicamente envía / recibe información de actualización sobre interfaces que pertenecen ha dicha red.
  NETWORK = [Prefijo de dirección de red IPv6 / Longitud Prefijo]

  IFNAME, RIPng envía / recibe información de actualización sobre una interfaz *IFNAME* específica.
  IFNAME = eth0, eth1 [Interfaces Ethernet]
  IFNAME = sitX, X>0 [Interfaz virtual Túnel IPv6 en IPv4]

- Active el proceso de enrutamiento RIPng sobre una interfaz *IFNAME* específica.

  ripngd-ipv6*(config-router)# no passive-interface IFNAME*

- Guarde los cambios y salga del modo de configuración de enrutador.

  ripngd-ipv6*(config-router)# write memory*
  ripngd-ipv6*(config-router)# end*

ripngd-ipv6*#disable*
ripngd-ipv6*>*

Habilitando RIPng - modo pasivo

- Habilite los privilegios del *modo EXEC*

  ripngd-ipv6*>enable*
  Password: ipv6
  ripngd-ipv6*#*

- Acceda al modo de configuración general

  ripngd-ipv6# *configure terminal*
  ripngd-ipv6 *(config)#*

- Acceda al *modo de configuración de enrutador.* De esta manera activa el proceso de enrutamiento RIPng.

  ripngd-ipv6*(config)# router ripng*
  ripngd-ipv6*(config-router)#*

- Active una interfaz *IFNAME* específica en modo pasivo.

  ripngd-ipv6*(config-router)# passive-interface IFNAME*

- Guarde los cambios y salga del modo de configuración de enrutador.

  ripngd-ipv6*(config-router)# write memory*
  ripngd-ipv6*(config-router)# end*
  ripngd-ipv6*#disable*
  ripngd-ipv6*>*

Temporizar RIPng

Los enrutadores que habilitan RIPng, pueden ajustar varios temporizadores de RIPng utilizando el comando ***timers basic*** en el *modo de configuración de enrutador*.

- Habilite los privilegios del *modo EXEC*

  ripngd-ipv6*>enable*
  Password: ipv6
  ripngd-ipv6*#*

- Acceda al modo de configuración general

  ripngd-ipv6# *configure terminal*
  ripngd-ipv6*(config)#*

- Acceda al *modo de configuración de enrutador.* De esta manera activa el proceso de enrutamiento RIPng.

  ripngd-ipv6*(config)# router ripng*
  ripngd-ipv6*(config-router)#*

- Ajuste los temporizadores de RIPng.

  ripngd-ipv6*(config-router)# timers basic update timeout garbage*
  ripngd-ipv6*(config-router)# timers basic <0-65535> <0-65535> <0-65535>*

  Nota: Los valores por defecto de los temporizadores RIPng son los siguientes:

  update - *temporizador de actualización*: valor por defecto, 30 segundos.
  Cada 30 segundos, el proceso de enrutamiento RIPng despierta y envía un mensaje de respuesta no solicitado que contiene la tabla de enrutamiento completa a todos los enrutadores vecinos.

  timeout - *temporizador de espera*: valor por defecto, 180 segundos.
  Si el temporizador de actualización expira sin recibirse un mensaje de actualización de los enrutadores vecinos, la ruta se considera inválida. Sin embargo, la ruta se retiene por un período de tiempo *timeout* hasta eliminarse definitivamente de la tabla de enrutamiento IPv6.

  garbage - *temporizador recolector de basura*: valor por defecto, 120 segundos.
  Hasta que el *garbage* expire, la ruta considerada inválida es incluida en los mensajes de actualización anunciados por el enrutador.

- Guarde los cambios y salga del modo de configuración de enrutador.

ripngd-ipv6*(config-router)# write memory*
ripngd-ipv6*(config-router)# end*
ripngd-ipv6*#disable*
ripngd-ipv6*>*

Redistribuir BGP4+ en RIPng

- Habilite los privilegios del *modo EXEC*

ripngd-ipv6*>enable*
Password: ipv6
ripngd-ipv6*#*

- Acceda al modo de configuración general

ripngd-ipv6# *configure terminal*
ripngd-ipv6 *(config)#*

- Acceda al *modo de configuración de enrutador.* De esta manera activa el proceso de enrutamiento RIPng.

ripngd-ipv6*(config)# router ripng*
ripngd-ipv6*(config-router)#*

- Redistribuya información BGP4+ en RIPng

ripngd-ipv6*(config-router)# redistribute bgp*

- Guarde los cambios y salga del modo de configuración de enrutador.

ripngd-ipv6*(config-router)# write memory*
ripngd-ipv6*(config-router)# end*
ripngd-ipv6*#disable*
ripngd-ipv6*>*

Verificación de la configuración y operación de RIPng

La verificación de la configuración de RIPng puede llevarse a cabo por medio de los siguientes comandos *EXEC show*:

- Despliegue las entradas de la tabla de enrutamiento RIPng
  ripngd-ipv6>
  ripngd-ipv6> *show ipv6 ripng*

4.2.5 **Recursos**

- 5 PC's con S.O Red Hat Linux (Enrutadores IPv6)
- 4 PC's con S.O Red Hat Linux o S.O Microsoft Windows 2000 (Hosts IPv6)
- 4 Hubs Ethernet
- Proveedor de servicios IPv6 (6bone ISP): UNAM
- Espacio de direcciones IPv6 (*bloque* pNLA /48): 3FFE:8070:1024::/48

4.2.6 **Desarrollo de la Práctica**

Esta práctica se divide en 2 fases:

- Fase 1: Intercambio de mensajes en RIPng
- Fase 2: Coexistencia entre RIPng como IGP y BGP4+ como EGP en el SA UniCauca IPv6

Fase 1: Intercambio de mensajes en RIPng

La Figura 44, describe la topología de la red piloto UniCauca IPv6 que se debe implementar en la Fase 1.

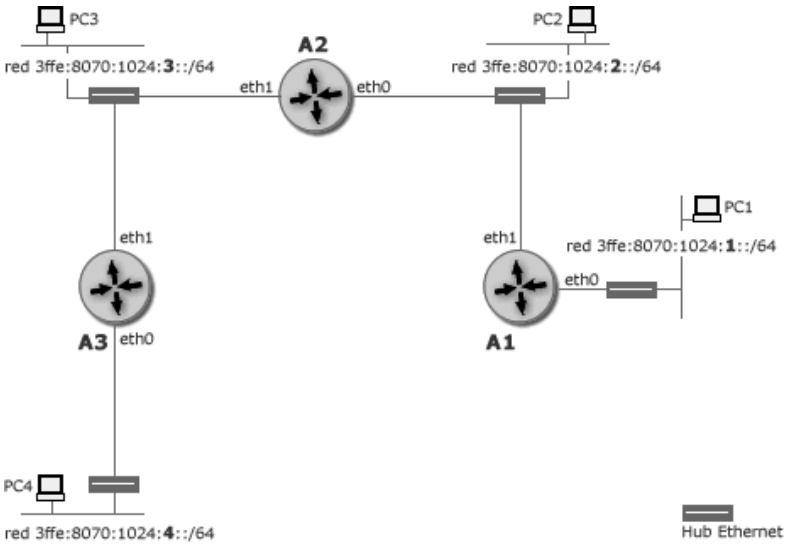


Figura 44 - Topología de la Fase 1

La Tabla 11, describe las direcciones IPv6 del tipo Unicast Global Agregable asignadas a las interfaces Ethernet de los enrutadores del SA UniCauca IPv6.

Enrutador Linux	**A1**	**A2**	**A3**
Interfaz Ethernet [eth0]	3FFE:8070:1024:1::**10**/64	3FFE:8070:1024:2::**20**/64	3FFE:8070:1024:4::**30**/64
Interfaz Ethernet [eth1]	3FFE:8070:1024:2::**10**/64	3FFE:8070:1024:3::**20**/64	3FFE:8070:1024:3::**30**/64

Tabla 11 - Direcciones IPv6 de los Enrutadores

Notas:

- Referirse a la Sección 3.3.2: Soporte IPv6 en RH Linux
- Referirse a la Sección 3.3.3: Configuración IPv6 de red en RH Linux

Los hosts **PC1** y **PC4** que pertenecen a las redes 3FFE:8070:1024:**1**::/64 y 3FFE:8070:1024:**4**::/64 respectivamente, fijan su dirección IPv6 por el proceso de *autoconfiguración sin control de estado*, a partir de su Identificador de Interfaz y del prefijo de red anunciado por el enrutador **A1**.

Los hosts **PC2** y **PC3** que pertenecen a las redes 3FFE:8070:1024:**2**::/64 y 3FFE:8070:1024:**3**::/64 respectivamente, fijan su dirección IPv6 por el proceso de *autoconfiguración sin control de estado*, a partir de su Identificador de Interfaz y de los prefijos de red anunciados por el enrutador **A2**.

4.2.7 **Procedimiento Fase 1**

Este procedimiento se basa en cada una de las tareas presentadas anteriormente para la configuración básica de RIPng en un equipo enrutador IPv6 utilizando *zebra*.

- En los enrutadores **A1**, **A2** y **A3**, habilite RIPng en modo activo. Utilizando los siguientes criterios de configuración :

  - Enrutador A1 anuncia su red por la interfaz eth1.
  - Enrutador A2 anuncia sus redes por las interfaces eth1 y eth0.
  - Enrutador A3 anuncia su red por la interfaz eth1.

- Verifique la configuración RIPng en modo activo utilizando el comando: ***ripngd-ipv6***> *show running-config.*

- Introduzca en los enrutadores **A1**, **A2** y **A3** el comando: ***ripngd-ipv6***> *show ipv6 ripng*, para examinar la tabla de enrutamiento RIPng de estos enrutadores.

P.4.2.1 Desde los enrutadores **A1**, **A2** y **A3** ¿Cuál es el Next Hop y la métrica para alcanzar las redes 3FFE:8070:1024:**1**::/64, 3FFE:8070:1024:**2**::/64, 3FFE:8070:1024:**3**::/64 y 3FFE:8070:1024:**4**::/64?

- Verifique lo anterior ejecutando el comando *traceroute6* desde los enrutadores **A1**, **A2** y **A3** hacia interfaces de red Ethernet en cada una las redes que pueden alcanzar estos enrutadores. (Ejemplo práctico, desde A1 a la Interfaz de red 3FFE:8070:1024:**4**::30: *traceroute6 3FFE:8070:1024:**4**::30*)

P.4.2.2 ¿Los resultados obtenidos en cada caso son consecuentes con las tablas de enrutamiento RIPng de los enrutadores **A1**, **A2** y **A3**?

- Utilice el analizador de protocolos de red Ethereal[60] para identificar el formato de mensajes RIPng que envía y recibe el enrutador **A2**. Para esto ejecute desde la consola de Linux de **A2** el comando ethereal, y reinicie el servicio RIPng utilizando el comando: *#service ripngd restart*

P.4.2.3 Describa el formato y los tipos de mensajes RIPng.

P.4.2.4 Describa el proceso de encapsulación de RIPng para su transporte sobre UDP/IPv6.

- Reconfigure la topología de red como se muestra en la Figura 45.

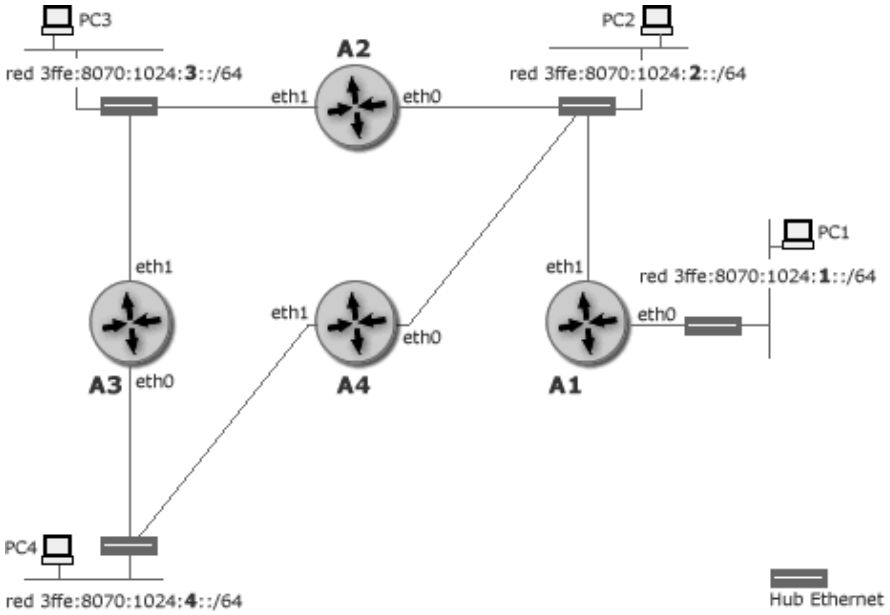


Figura 45 - Topología de la Fase 1: Reconfiguración de la topología de red

[60] Referirse a la sección 3.3.1: Instalación de Red Hat Linux 9 y de las aplicaciones utilizadas en el Laboratorio Internet IPv6. Se recomienda filtrar el protocolo RIPng. Información detallada acerca del funcionamiento de *ethereal* se encuentra en: http://www.ethereal.com/docs/

La Tabla 12, describe las direcciones IPv6 del tipo Unicast Global Agregable asignadas a las interfaces Ethernet del enrutador **A4** del sistema autónomo UniCauca IPv6.

Enrutador Linux	**A4**
Interfaz Ethernet [eth0]	3FFE:8070:1024:2::**40**/64
Interfaz Ethernet [eth1]	3FFE:8070:1024:4::**40**/64

Tabla 12 - Direcciones IPv6 del Enrutador A4

- En el enrutador **A4**, habilite RIPng en modo pasivo.

- Verifique la configuración actual de RIPng: ***ripngd-ipv6****# show running-config*

- Ejecute en los enrutadores **A1**, **A2**, **A3** y **A4** el comando: ***ripngd-ipv6****> show ipv6 ripng*, para examinar la tabla de enrutamiento RIPng de estos enrutadores.

P.4.2.5 ¿Encontró cambios en las tablas de enrutamiento RIPng de **A1**, **A2** y **A3**? Justifique su respuesta.

P.4.2.6 Llene la siguiente tabla:

	A1		**A2**		**A3**		**A4**	
	Next Hop	Métrica	Next Hop	Métrica	Next Hop	Métrica	Next Hop	Métrica
3FFE:8070:1024:**1**::/64								
3FFE:8070:1024:**2**::/64								
3FFE:8070:1024:**3**::/64								
3FFE:8070:1024:**4**::/64								

Tabla 13 - Información tablas de enrutamiento RIPng

- Ahora en el enrutador **A4**, habilite RIPng en modo activo.

- Ejecute en los enrutadores **A1**, **A2**, **A3** y **A4** el comando: ***ripngd-ipv6****> show ipv6 ripng*, para examinar la tabla de enrutamiento RIPng de estos enrutadores.

P.4.2.7 ¿Encontró cambios en las tablas de enrutamiento RIPng de **A1**, **A2** y **A3**? Justifique su respuesta.

P.4.2.8 ¿Cuáles son las entradas en la tabla de enrutamiento del enrutador **A1**?

P.4.2.9 ¿Qué se puede concluir del funcionamiento de RIPng en modo pasivo y en modo activo?

- Desconecte la interfaz eth0 en el enrutador **A4**. De esta forma se da de baja el enlace entre las redes 3FFE:8070:1024:**2**::/64 y 3FFE:8070:1024:**4**::/64.

- Ejecute en el enrutador **A1** el comando: ***ripngd-ipv6***> *show ipv6 ripng* varias veces, para examinar la tabla de enrutamiento RIPng .

P.4.2.10 ¿Encontró cambios en las tabla de enrutamiento RIPng de **A1**? Justifique su respuesta.

- Conecte nuevamente la interfaz eth0 en el enrutador **A4**. Ejecute en el enrutador **A1** el comando: ***ripngd-ipv6***> *show ipv6 ripng*, para examinar la tabla de enrutamiento RIPng de este enrutador.

P.4.2.11 ¿Encontró cambios en las tabla de enrutamiento RIPng de **A1**? Justifique su respuesta.

P.4.2.12 ¿Cuál es su concepto sobre la convergencia de RIPng en los eventos anteriores?

- Temporice RIPng en los 4 enrutadores asignando los siguientes valores: *update=5*, *timeout=30*, *garbage=20*

- Repita el procedimiento de los 2 casos anteriores (desconectar y conectar eth0 en **A4** y examine la tabla de enrutamiento RIPng en **A1**).

P.4.2.13 ¿Encontró cambios en las tabla de enrutamiento RIPng de **A1**? ¿Cuáles son sus conclusiones?

- Reconfigure la topología de red como se muestra en la Figura 46.

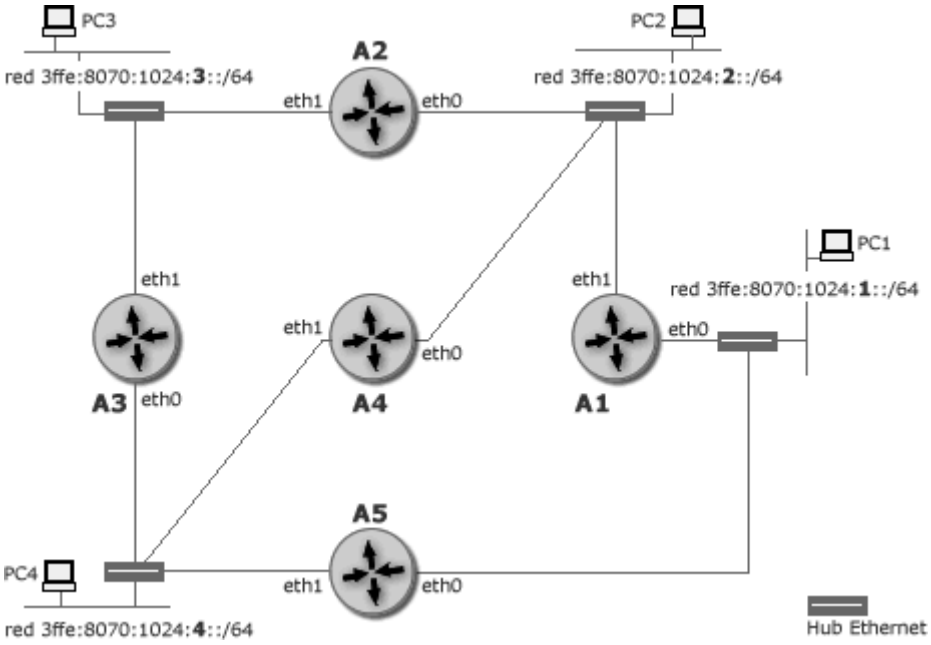


Figura 46 - Topología de la Fase 1: Reconfiguración de la topología de red

La Tabla 14, describe las direcciones IPv6 Unicast Globales Agregables asignadas a las interfaces Ethernet del enrutador **A4** del SA UniCauca IPv6.

Enrutador Linux	**A5**
Interfaz Ethernet [eth0]	3FFE:8070:1024:1::**50**/64
Interfaz Ethernet [eth1]	3FFE:8070:1024:4::**50**/64

Tabla 14 - Direcciones IPv6 del Enrutador A5

- En el enrutador **A5**, habilite RIPng en modo activo.

- Ejecute en los enrutadores **A3** y **A5** el comando: ***ripngd-ipv6***> *show ipv6 ripng*, para examinar la tabla de enrutamiento RIPng de estos enrutadores.

P.4.2.14 ¿Qué observa en **A3** respecto a la red 1 y en **A5** respecto a la red 3?

P.4.2.15 De acuerdo a la información de las tablas de enrutamiento en **A1**, **A2**, **A3**, **A4** y **A5** llene la siguiente tabla

	A1		**A2**		**A3**		**A4**		**A5**	
	Next Hop	Métrica	Next Hop	Métrica	Next Hop	Métrica	Next Hop	Métrica	Next Hop	Métrica
3FFE:8070:1024:**1**::/64										
3FFE:8070:1024:**2**::/64										
3FFE:8070:1024:**3**::/64										
3FFE:8070:1024:**4**::/64										

Tabla 15 - Información tablas de enrutamiento RIPng

- Verifique lo anterior ejecutando el comando *traceroute6* desde los enrutadores **A1**, **A2**, **A3**, **A4** y **A5** hacia interfaces de red Ethernet en cada una las redes IPv6 que pueden alcanzar estos enrutadores. (Ejemplo práctico, desde A1 a la Interfaz de red 3FFE:8070:1024:**4**::30 en la red 3FFE:8070:1024:**4**::/64 - *traceroute6 3FFE:8070:1024:**4**::30*)

P.4.2.16 ¿Los resultados obtenidos en cada caso, son consecuentes con los datos de la tabla anterior?

Fase 2: Coexistencia entre RIPng como IGP y BGP4+ como EGP en el SA UniCauca IPv6

La Figura 47, describe la topología de la red piloto UniCauca IPv6 que se debe implementar en la Fase 2. Esta fase retoma la configuración RIPng de la topología inmediatamente anterior y facilita la salida a 6bone del SA UniCauca IPv6 al permitir coexistir RIPng y BGP4+.

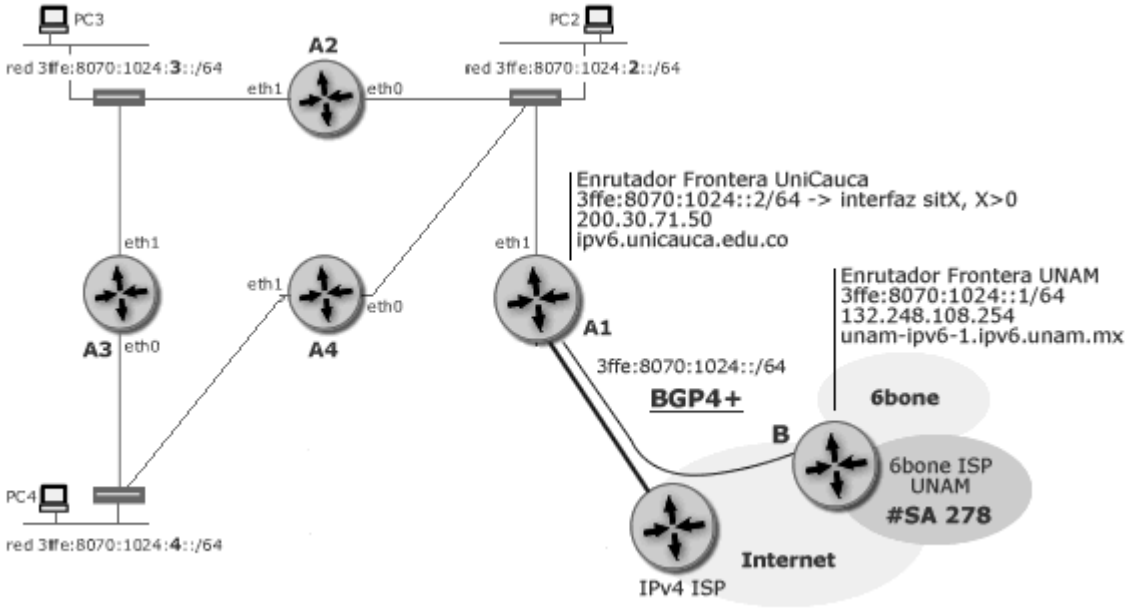


Figura 47 - Topología de la Fase 2

4.2.8 **Procedimiento Fase 2**

Este procedimiento se basa en cada una de las tareas presentadas anteriormente para la configuración básica de RIPng y BGP4+ en un equipo enrutador IPv6 utilizando *zebra*.

- En el enrutador **A1**, configure el proceso de enrutamiento BGP4+ utilizando el #SA 64515[61].

[61] Referirse a la sección: 4.1.4.1 Implementación del multiprotocolo BGP4+ en Red Hat Linux

- En el enrutador **A1**, configure iguales o vecinos BGP4+, para establecer al enrutador **B** 3FFE:8070:1024::1/64 como igual o vecino BGP4+.

- En el enrutador **A1**, anuncie rutas a través de BGP4+ para publicar el espacio de direcciones IPv6 3FFE:8070:1024::/48 al SA UNAM IPv6.

- Verifique la configuración actual de BGP4+: ***bgpd-ipv6****# show running-config*

- Verifique la configuración actual de RIPng: ***ripngd-ipv6****# show running-config*

- En el enrutador **A1**, redistribuya BGP4+ en RIPng ejecutando el siguiente comando: ***ripngd-ipv6(config-router)#*** *redistribute bgp*

- Ejecute en los enrutadores **A1**, **A2**, **A3**, **A4** el comando: ***ripngd-ipv6***> *show ipv6 ripng*, para examinar la tabla de enrutamiento RIPng de estos enrutadores.

P.4.2.17 ¿Analizando las tablas de enrutamiento RIPng y BGP4+ que concluye respecto al Next Hop en cada caso?

- Verifique lo anterior ejecutando el comando *traceroute6* desde los enrutadores **A1**, **A2**, **A3**, **A4** hacia el extremo *remoto* del túnel, enrutador principal del sitio UNAM IPv6 y a un equipo cualquiera en 6bone, por ejemplo 3FFE:B00:C18:1::10. Para esto ejecute los comandos #*traceroute6 3ffe:8070:1024::1* y # *traceroute6 3ffe:b00:c18:1::10*

P.4.2.18 ¿Esta prueba de alcanzabilidad es consecuente con las tablas de enrutamiento RIPng de los enrutadores **A1**, **A2**, **A3** y **A4**?

4.2.9 **Análisis de Resultados**

4.2.10 **Conclusiones**

4.3 Protocolo OSPF con soporte a IPv6 - OSPFv6

4.3.1 Motivación

Esta práctica explica como explorar las capacidades del protocolo OSPFv6 - *terminología, funcionamiento, configuración, verificación* - sobre diversos escenarios de enrutamiento propuestos para la red piloto UniCauca IPv6. Para esto se trabajará en 3 fases: Intercambio de mensajes OSPFv6, Múltiples caminos OSPFv6, y Coexistencia entre OSPFv6 como IGP y BGP4+ como EGP en el SA UniCauca IPv6.

4.3.2 Objetivos

- Habilitar y configurar OSPFv6 como un IGP en el conjunto de enrutadores que forman la red piloto UniCauca IPv6 para intercambiar información de enrutamiento al interior del sistema autónomo UniCauca IPv6.

- Redistribuir BGP4+ en OSPFv6 para estudiar las capacidades que tienen los enrutadores fronterizos de conectar distintos sistemas autónomos con el fin de intercambiar y publicar información de enrutamiento recibida de un sistema autónomo a otro sistema autónomo.

- Verificar la conectividad en el sistema autónomo UniCauca IPv6 y hacia 6bone.

4.3.3 Marco teórico

El *Open Shortest Path First*, primero la ruta libre más corta (OSPF), es un protocolo de enrutamiento de estado de enlace en contraposición a los protocolos de enrutamiento por vector distancia como el *Routing Information Protocol,* Protocolo de información de enrutamiento (RIP). OSPF es un *Interior Gateway Protocol* , Protocolo de gateway interior (IGP) que en su versión 3 extiende las funcionalidades de OSPFv2 para soportar la arquitectura de direccionamiento IPv6 y dar respuesta a las necesidades de las redes IPv6 escalables que RIPng no puede afrontar.

La información de OSPFv6[62] se transporta dentro de paquetes IPv6, utilizando el número de protocolo 89.

[62] OSPF para IPv6, versión 3 del protocolo de estado de enlace OSPF. Es un estándar abierto que está disponible en la información pública, la especificación de OSPF para IPv6 es publicada como un Request For Comments (RFC) 2740.

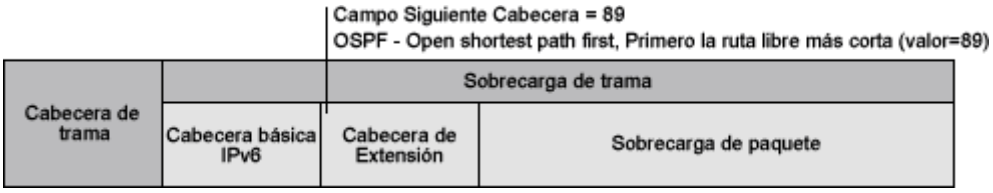


Figura 48 - OSPFv6 en un paquete IPv6

4.3.3.1 **Funcionamiento de OSPFv6**

OSPFv6 requiere que cada enrutador participante tenga información de la topología de la red IPv6 completa, y dado que el enrutamiento depende del estado de los enlaces entre enrutadores, los enrutadores vecinos deben reconocerse entre sí en la red antes de poder compartir información. Este proceso se hace por medio de *mensajes hello*, que se envían periódicamente a cada interfaz de los enrutadores que están participando en OSPFv6, con el fin de establecer y mantener relaciones de vecindad garantizando la comunicación entre vecinos.

OSPFv6 sólo genera *link state updates*, actualizaciones de estado de enlace (LSU)[63] cuando hay cambios en la topología de la red. Cuando un enlace cambia de estado, el enrutador que detecta el cambio activa un *link state advertisements*, publicación de estado de enlace (LSA) con información sólo del enlace que ha cambiado (como una ruta individual), y luego esta LSA se propaga a los enrutadores vecinos. Cada enrutador toma una copia de la LSA, la reenvía a todos los enrutadores vecinos (este proceso se denomina *flooding*, o técnica de inundación) y actualiza su base de datos de estado de enlace. Esta inundación de LSA's garantiza a todos los enrutadores conocer el cambio de estado del enlace con el fin de actualizar sus bases de datos y generar a partir del algoritmo *Shortest Path First* (SPF) una tabla de enrutamiento que refleje la nueva topología[64]. Este enfoque ahorra ancho de banda en cada enlace, ya que las publicaciones contienen menos información que una tabla de enrutamiento IPv6 completa y sólo se envían cuando hay un cambio en la topología de la red.

Figura 49 - OSPFv6 envía entradas individuales IPv6 a sus vecinos

[63] Los mensajes LSU implementan la técnica de inundación a través de LSA's.
[64] La base de datos del estado de enlace contiene una lista completa de entradas del estado de enlace de todos los enrutadores en la red, así que todos los enrutadores dentro de un área tienen bases de datos idénticas. La tabla de enrutamiento contiene la lista *Shortest Paths*, rutas más cortas a destinos conocidos vía enrutadores específicos. El contenido de cada tabla de enrutamiento es único.

4.3.3.2 **Enrutador designado y enrutador designado de reserva**

OSPFv6 evita la sobrecarga de información de enrutamiento en entornos IPv6 de acceso múltiple[65] donde existe un número significativo de enrutadores, al centralizar la información de enrutamiento en dos enrutadores: un *Designated Router*, enrutador designado (DR) y un *Backup Designated Router*, enrutador designado de reserva (BDR) para que representen la red. El BDR no lleva a cambio ninguna función DR cuando este último esta funcionando. En su lugar, recibe toda la información, pero permite al DR llevar a cabo las tareas de reenvío y sincronización. El BDR sólo lleva a cabo tareas DR si éste falla.

El DR y el BDR añaden valor a la red de las siguientes formas:

- *Disminuyendo el tráfico de actualización de enrutamiento*: El DR y el BDR actúan como punto de contacto para el intercambio de información de estado de enlace. Por lo tanto, cada enrutador participe en OSPFv6 debe establecer una adyacencia con el BR y el BDR. En vez de que cada enrutador intercambie información de estado de enlace con los demás enrutadores vecinos, cada enrutador envía esta información al BR y al BDR. El DR representa la red en el sentido que envía información de estado de enlace de cada enrutador a todos los demás enrutadores de la red.

- *Manipulando la sincronización del estado de enlace*: El DR y BDR aseguran que los demás enrutadores de la red tengan la misma información de estado de enlace.

Para elegir un DR y un BDR, los enrutadores ven el valor de prioridad de cada uno durante el proceso de intercambio de mensajes *hello*. Luego utilizan las condiciones siguientes para determinar cuál de ellos se elige:

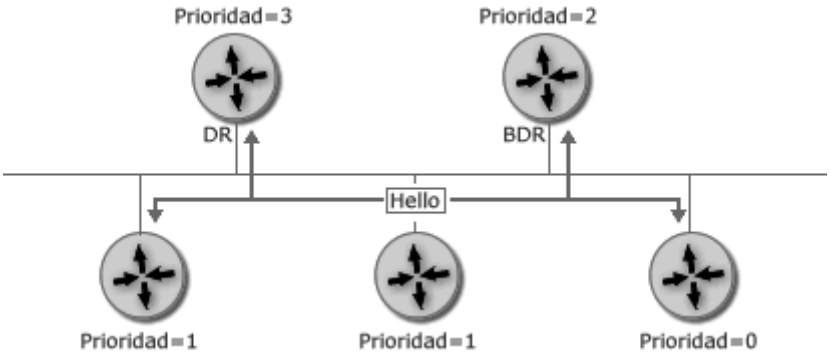


Figura 50 - Elección de DR y BDR

[65] Entre las topologías IPv6 de acceso múltiple se tiene: topologías que soportan más de dos enrutadores, con capacidad de difusión para transmitir mensajes a todos los enrutadores (Ej. redes Ethernet), y topologías que soportan más de dos enrutadores, pero que no tienen capacidad de difusión, NoBroadcast MultiAccess NBMA (Ej. redes ATM, X.25 y Frame Relay).

- El enrutador que tiene el valor de prioridad más alto es el DR.
- El enrutador que tiene el segundo valor de prioridad es el BDR.
- El valor predeterminado de la prioridad OSPFv6 de una interfaz es 1. En caso de igualdad, se usa el ID[66] de enrutador.
- Un enrutador con un conjunto de prioridades *0* no es elegible para convertirse en un DR o un BDR.

Si un enrutador con un valor de prioridad más alto se añade a la red, el DR y el BDR no cambian. Un DR o un BDR cambian sólo si uno de ellos se cae. Si es el DR el que cae, el BDR se convierte en el DR, y se elige un nuevo BDR. Si es el BDR el que cae, se elige un nuevo BDR.

4.3.3.3 **Múltiples áreas OSPFv6**

OSPFv6 se vale del concepto de enrutamiento jerárquico y permite dividir un dominio IPv6 en múltiples áreas lógicas que pueden intercambiar información de enrutamiento y que se identifican por direcciones de 128bits, de este modo el enrutamiento sigue produciéndose entre las áreas pero un enrutador solo necesita conocer la topología e información de enrutamiento correspondiente a su área. El planteamiento jerárquico reduce la necesidad de inundar una LSA en todo el dominio de enrutamiento IPv6, ya que el uso de áreas restringe la inundación al límite lógico del área y un cambio en el área ocasiona el recálculo de la tabla de enrutamiento para los enrutadores de sólo esa área, no para todo el dominio.

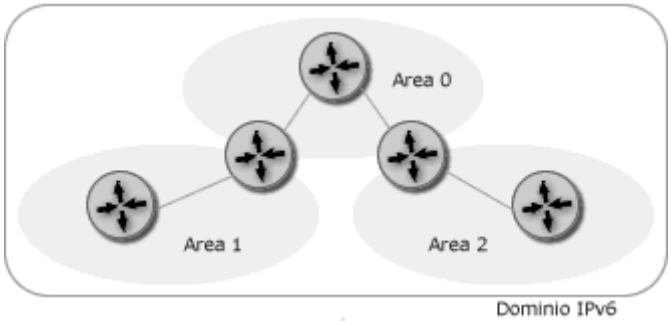


Figura 51 - Enrutamiento entre múltiples áreas IPv6

La topología jerárquica de OSPF posee las siguientes ventajas:

- Frecuencia reducida de los cálculos SPF
- Tablas de enrutamiento más pequeñas
- Estructura reducida del *link state update*, actualización de estado del enlace LSU.

[66] ID de enrutador: Identificador único de enrutador dentro de un sistema autónomo. La dirección IPv6 más alta de la interfaz activa se elige por defecto.

4.3.3.4 **Tipos de Enrutadores OSPF IPv6**

- *Intern Router*, enrutador interno: tiene todas sus interfaces en la misma área. Los enrutadores *internos* que estén en la misma área poseen bases de datos de estado de enlace idénticas.

- *Backbone Router*, enrutador backbone: tiene al menos una interfaz conectada al *área 0* denominada *backbone*. Estos enrutadores mantienen información de enrutamiento OSPFv6 utilizando los mismos procedimientos y algoritmos que los enrutadores internos. El *área 0* sirve como área de tránsito entre las demás áreas OSPFv6.

- *Area Border Router,* enrutador fronterizo de área (ABR): tiene sus interfaces conectadas a múltiples áreas y por tanto las interconectan. Estos enrutadores mantienen bases de datos de estado de enlace separadas en cada área a la que están conectados, y enrutan el tráfico entre áreas. Los ABR pueden resumir información de sus bases de datos de estado de enlace de sus áreas conectadas y distribuir la información al área *backbone*, luego reenviar la información a todas las demás áreas conectadas. Un área puede tener uno o más ABR.

- *Autonomous System Boundary Router,* enrutador fronterizo de sistema autónomo (ASBR): tiene al menos una interfaz para intercambiar tráfico con enrutadores de otro sistema autónomo. Estos enrutadores pueden *redistribuir* (importar) información de red no OSPFv6 a la red OSPFv6 y viceversa.

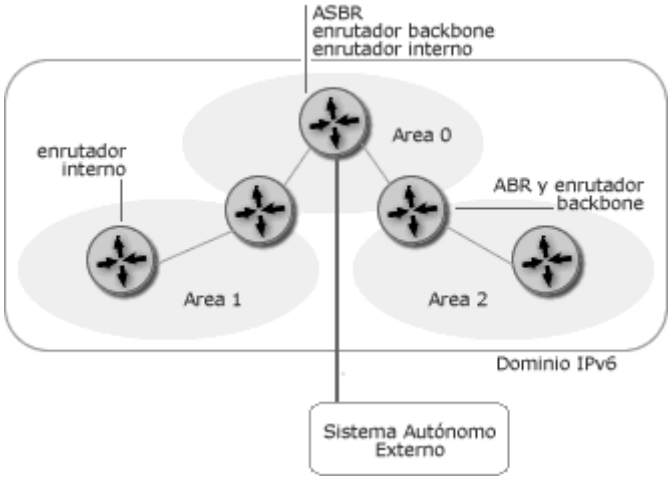


Figura 52 **-** Tipos de enrutadores OSPF IPv6

Un enrutador posee una base de datos de estado de enlace separada por cada área a la que está conectado. Por tanto, un ABR tendría una base de datos de estado de enlace para el *área 0* y otra base de datos de estado de enlace para la otra área en la que participa. Dos enrutadores que pertenecen a la misma área tienen, para esa área, bases de datos de estado

de enlace de área idénticas. Una base de datos de estado de enlace se sincroniza entre pares de enrutadores adyacentes, lo que significa que está sincronizada entre un enrutador y su enrutador designado (DR) y su enrutador de reserva (BDR).

4.3.3.5 **Tipos de LSA IPv6**

- Tipo 1: *Router LSA*, generada por cada uno de los enrutadores que pertenecen a un área determinada. Esta LSA describe el estado y costo de las interfaces de un enrutador y muestra si un enrutador es un ABR ó un ASBR.

- Tipo 2: *Network LSA*, generada por enrutadores designados (DR) en redes de acceso múltiple. Esta LSA describe los enrutadores conectados a la red, y no provee información de atributos de direcciones IPv6.

- Tipo 3: *Interarea-prefix LSA for ABR's*, generada por enrutadores fronterizos de área (ABR). Esta LSA anuncia redes internas a enrutadores que pertenecen a otras áreas, y define direcciones IPv6 como un prefijo de dirección IPv6 y una longitud del prefijo.

- Tipo 4: *Interarea-router LSA for ASBR's*, generada por enrutadores fronterizos de sistema autónomo (ABR). Esta LSA anuncia la localización de un ASBR en un SA.

- Tipo 5: *Autonomous System external LSA*, generada por enrutadores fronterizos de sistema autónomo (ASBR). Esta LSA anuncia las rutas a los destinos que son externos al sistema autónomo, y define direcciones IPv6 como un prefijo de dirección IPv6 y una longitud del prefijo. La ruta por defecto es expresada como un prefijo de longitud 0.

- Tipo 8: *Link LSA*, técnica de inundación de ámbito local. Esta LSA provee direcciones IPv6 de enlace local a todos los enrutadores que pertenecen al enlace, y describe la lista de prefijos de direcciones IPv6 asociados al enlace.

- Tipo 9: *Intra-Area-Prefix LSA*, asocia una lista de prefijos de direcciones IPv6 con un enlace a una *transit network*[67] tomando como referencia una *Network LSA*, o asocia una lista de prefijos de direcciones IPv6 con un enlace a un *enrutador* tomando como referencia una *Router LSA.*

[67] Red de Tránsito

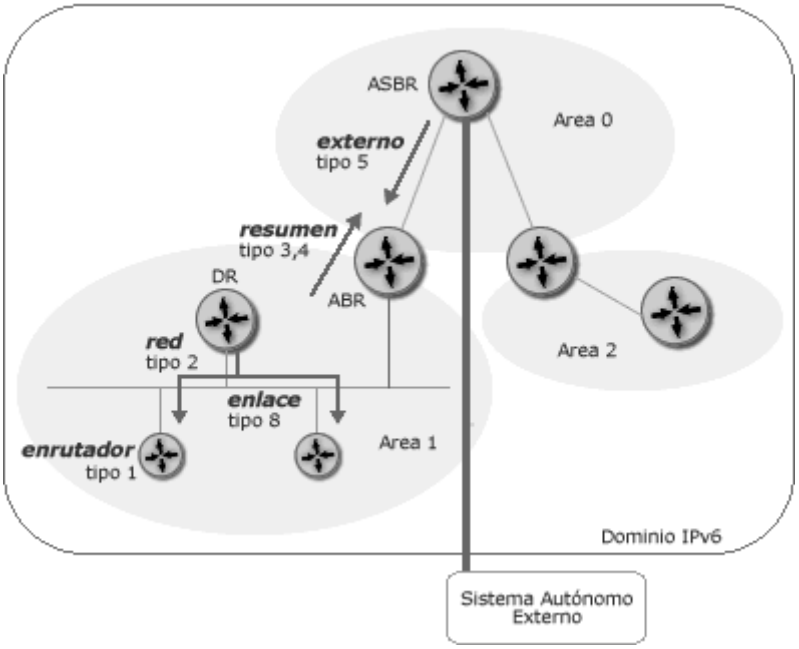


Figura 53 - LSA's inundadas en una red OSPFv6

4.3.4 **Desarrollo técnico**

4.3.4.1 **Implementación de OSPFv6 en Red Hat Linux**

Para implementar OSPFv6 en Red Hat Linux se utilizará *zebra*[68], el cual es un software de enrutamiento avanzado distribuido bajo Licencia Pública GNU (GPL) que administra diferentes protocolos de enrutamiento basados en TCP/IP.

Nota: Desde una consola de Linux ejecute el comando *# rpm –qa | grep zebra*, para verificar la instalación del paquete *zebra* en el equipo de cómputo. Si *zebra* no está instalado, los CD-ROM's de RH Linux 9 facilitan este paquete en formato RPM, para esto ejecute el comando:

 # rpm –i zebra-0.93b-1.i386.rpm

Arquitectura del sistema zebra - OSPFv6

Zebra es un software modular, donde cada protocolo de enrutamiento es un servicio independiente y es gestionado por un servicio principal llamado *zebra*, el kernel gestor del enrutamiento[69]. La arquitectura de sistema *zebra* para OSPFv6 se muestra a continuación:

[68] Otras implementaciones: GateD*****, MRTd*****, Kame BGPd [***** están bajo desarrollo]
[69] La modularidad que ofrece *zebra* permite que cada servicio tenga su propio archivo de configuración e Interfaz de terminal.

Figura 54 - Arquitectura del Sistema Zebra

Trabajando con zebra - OSPFv6

En el directorio */etc/zebra/* se deben crear los archivos de configuración *zebra.conf* y *ospf6d.conf*, necesarios para la configuración de estos 2 servicios.

- El archivo *zebra.conf* deberá tener las siguientes entradas:

```
hostname zebra-ipv6
password ipv6
enable password ipv6
log stdout
```

- El archivo *ospf6d.conf* deberá tener las siguientes entradas:

```
hostname ospf6d-ipv6
password ipv6
enable password ipv6
log stdout
```

- Inicie los servicios *zebra* y *ospf6d,* y acceda al archivo de configuración *ospf6d.conf* mediante una conexión *telnet* al puerto *2606* que corresponde a *ospf6d.* Puede acceder al servicio *ospf6d* desde el propio equipo (localhost) o bien desde equipos remotos.

[root@ipv6 /]# service zebra start [OK]
[root@ipv6 /]# service ospf6d start [OK]

*[root@ipv6 /]# telnet localhost **2606***
Trying 127.0.0.1...
Connected to localhost.

Hello, this is zebra (version 0.93b).
Copyright 1996-2002 Kunihiro Ishiguro.

User Access Verification
Password:

- Ingrese el Password que por defecto será "ipv6" y estará en el modo *EXEC* (comandos generales) del servicio OSPFv6.

  ospf6d-ipv6>

Nota: Cada una de las tareas descritas a continuación se identifica como requerida u opcional en la implementación de OSPFv6.

Tareas:

- Habilitando OSPFv6 en una interfaz *(requerida)*
- Configurando atributos OSPFv6 *(opcional)*
- Redistribuir BGP4+ en OSPFv6 *(opcional)*
- Verificación de la configuración y operación en OSPFv6 *(opcional)*

Habilitando OSPFv6 en una interfaz

Prerrequisitos[70]: Antes de configurar los enrutadores para que corran OSPFv6, confirme que las interfaces que anunciarán las redes dentro del SA UniCauca dispongan de una dirección IPv6 Unicast Global Agregable, y habilite el *reenvió* (forwarding) de tráfico global IPv6 en los enrutadores[71].

- Habilite los privilegios del *modo EXEC*

  ospf6d-ipv6>*enable*
  Password: ipv6
  ospf6d-ipv6#

- Acceda al modo de configuración general

  ospf6d-ipv6# *configure terminal*
  ospf6d-ipv6*(config)#*

[70] Para una mayor comprensión, referirse a la Sección 3.3.3: Configuración IPv6 de red en RH Linux.
[71] Verifique el parámetro IPV6FORWARDING=yes en el archivo */etc/sysconfig/network*

- Acceda al *modo de configuración de enrutador.* De esta manera activa el proceso de enrutamiento OSPFv6.

  ospf6d-ipv6*(config)# router ospf6*
  ospf6d-ipv6*(config-ospf6)#*

  Nota: *ospf6d* no soporta múltiples procesos OSPFv6. Por esta razón *ospf6d* no soporta múltiples áreas OPSFv6.

- Habilite el proceso de enrutamiento OSPFv6 sobre una interfaz *IFNAME* específica.

  ospf6d-ipv6*(config-ospf6)# interface IFNAME area A.B.C.D*

  IFNAME = eth0, eth0:0, eth1, eth1:0, eth1:1 [Interfaces Ethernet]
  IFNAME = sitX, X>0 [Interfaz virtual Túnel IPv6 en IPv4]
  A.B.C.D = [Identificador de *área* OSPFv6 en notación IPv4/32bits]

- Asigne un identificador ID de enrutador.

  ospf6d-ipv6*(config-ospf6)# router-id ROUTER-ID*

  ROUTER_ID = [Este número de 32 bits en notación IPv4/32bits identifica únicamente al enrutador dentro de un sistema autónomo. Esta identificación es importante a la hora de establecer relaciones de vecindad, además se usa para separar los enlaces durante el proceso de selección del enrutador designado DR y del enrutador designado de reserva BDR, siempre que los valores prioritarios sean iguales]

- Guarde los cambios y salga del modo de configuración de enrutador.

  ospf6d-ipv6*(config-ospf6)# write memory*
  ospf6d-ipv6*(config-ospf6)# end*
  ospf6d-ipv6*# disable*
  ospf6d-ipv6*>*

Configurando atributos OSPFv6 *(opcional)*

- Habilite los privilegios del *modo EXEC*

  ospf6d-ipv6*>enable*

Password: ipv6
ospf6d-ipv6*#*

- Acceda al modo de configuración general

ospf6d-ipv6# *configure terminal*
ospf6d-ipv6*(config)#*

- Acceda al *modo de configuración de interfaz*. Configure los atributos OSPFv6 *(cost, hello-interval, dead-interval, retransmit-interval, priority, transmit-delay)* sobre una interfaz *IFNAME* específica.

ospf6d-ipv6*(config)# interface IFNAME*
IFNAME = eth0, eth1 [Interfaces Ethernet]
IFNAME = sitX, X>0 [Interfaz virtual Túnel IPv6/IPv4]

ospf6d-ipv6*(config-if)#*
ospf6d-ipv6*(config-if)# ipv6 ospf6 cost COST*
COST = <1-65535> [Métrica asignada a la interfaz - El costo de ruta es el total de los costos asignados a todas las interfaces que reenvían tráfico hacía el destino - El valor predeterminado es 1. En interfaces Ethernet es 10]

ospf6d-ipv6 *(config-if)# ipv6 ospf6 hello-interval HELLO-INTERVAL*
HELLO-INTERVAL = <1-65535> [Intervalo *hello* - tiempo en segundos de la frecuencia con la que el enrutador envía mensajes *hello* - El valor predeterminado es 10 en redes de acceso múltiple]

ospf6d-ipv6 *(config-if)# ipv6 ospf6 dead-interval ROUTER-DEAD-INTERVAL*
ROUTER-DEAD-INTERVAL = <1-65535> [Intervalo *dead* - tiempo en segundos que el enrutador espera para oír a su vecino antes de declararlo caído. Por defecto, es cuatro veces el tiempo del Intervalo *hello*]

ospf6d-ipv6 *(config-if)# ipv6 ospf6 priority PRIORITY*
PRIORITY = <0-255> [Número que indica la prioridad del enrutador. Entre mayor sea la prioridad del enrutador, mayores serán las opciones de que sea elegido DR o BDR - El valor predeterminado es 1. Un valor de prioridad de 0 indica que no se puede elegir una interfaz como DR o BDR]

ospf6d-ipv6*(config-if)# ipv6 ospf6 retransmit-interval RXMTINTERVAL*

RXMTINTERVAL = <1-65535> [tiempo en segundos del temporizador de retransmisión - Este valor se utiliza para retransmitir *mensajes DataBase Description*, descripción de base de datos (DBD), que incluyen información de las cabeceras LSA tipo 3 y tipo 4 de las entradas LSA que aparecen en la base de datos del estado de enlace o sobre una red. Cada cabecera LSA incluye cosas como el tipo de estado de enlace, la dirección IPv6 del enrutador que publica y el número de secuencia LSA. Este número es la forma en que un enrutador determina la novedad de la información sobre el estado de enlace recibida. El DBD también incluye un número de secuencia DBD que asegura que se reciban todos los DBD en el proceso de sincronización de la base de datos. El valor predeterminado es 5]

- Guarde los cambios y salga del modo de configuración de enrutador.

ospf6d-ipv6*(config-if)# write memory*
ospf6d-ipv6*(config-if)# end*
ospf6d-ipv6*# disable*
ospf6d-ipv6*>*

Redistribuir BGP4+ en OSPFv6

- Habilite los privilegios del *modo EXEC*

ospf6d-ipv6*>enable*
Password: ipv6
ospf6d-ipv6*#*

- Acceda al modo de configuración general

ospf6d-ipv6# *configure terminal*
ospf6d-ipv6*(config)#*

- Acceda al *modo de configuración de enrutador.* De esta manera activa el proceso de enrutamiento OSPFv6.

ospf6d-ipv6*(config)# router ospf6*
ospf6d-ipv6*(config-ospf6)#*

- Redistribuya información BGP4+ en OSPFv6

ospf6d-ipv6*(config-ospf6)# redistribute bgp*

- Guarde los cambios y salga del modo de configuración de enrutador.

ospf6d-ipv6*(config-ospf6)# write memory*
ospf6d-ipv6*(config-ospf6)# end*
ospf6d-ipv6*# disable*
ospf6d-ipv6*>*

4.3.5 **Recursos**

- 4 PC's con S.O Red Hat Linux (Enrutadores IPv6)
- 4PC's con S.O Red Hat Linux o MS Windows 2000 (Hosts IPv6)
- 5 Hubs Ethernet
- Proveedor de servicios IPv6 (6bone ISP): UNAM
- Espacio de direcciones IPv6 (*bloque* pNLA /48): 3FFE:8070:1024::/48

4.3.6 **Desarrollo de la Práctica**

Esta práctica se divide en 3 fases:

- Fase 1: Intercambio de mensajes en OSPFv6
- Fase 2: Múltiples Caminos OSPFv6
- Fase 3: Coexistencia entre OSPFv6 como IGP y BGP4+ como EGP en el SA UniCauca IPv6

Fase 1: Intercambio de mensajes en OSPFv6

La Figura 55, describe la topología de la red piloto UniCauca IPv6 que se debe implementar en la Fase 1.

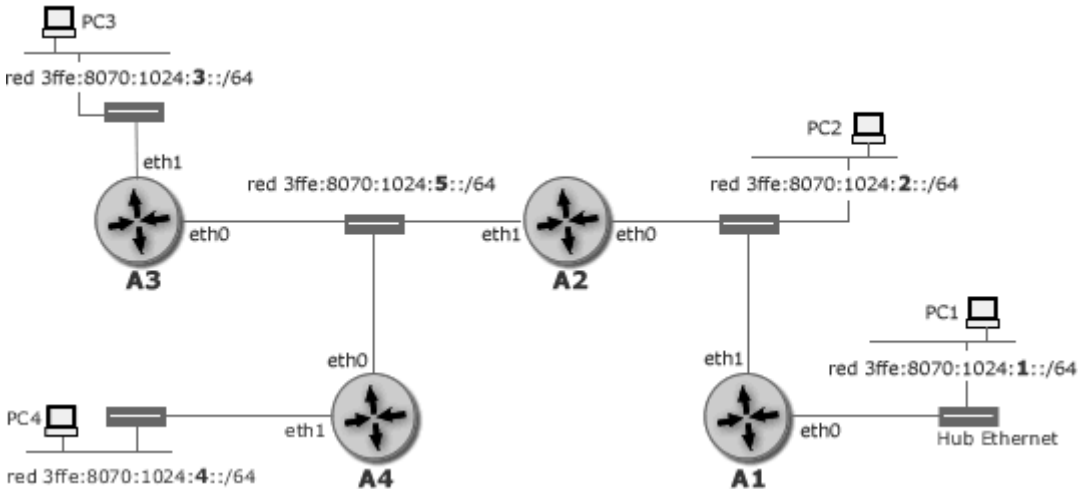


Figura 55 - Topología de la Fase 1

La Tabla 16, describe las direcciones IPv6 del tipo Unicast Global Agregable asignadas a las interfaces Ethernet de los enrutadores del SA UniCauca IPv6.

Enrutador	**A1**	**A2**	**A3**	**A4**
[eth0]	3FFE:8070:1024:1::**1**/64	3FFE:8070:1024:2::**2**/64	3FFE:8070:1024:5::**3**/64	3FFE:8070:1024:5::**4**/64
[eth1]	3FFE:8070:1024:2::**1**/64	3FFE:8070:1024:5::**2**/64	3FFE:8070:1024:3::**3**/64	3FFE:8070:1024:4::**4**/64

Tabla 16 - Direcciones IPv6 de los Enrutadores

Notas:

- Referirse a la Sección 3.3.2: Soporte IPv6 en RH Linux
- Referirse a la Sección 3.3.3: Configuración IPv6 de red en RH Linux

Los hosts **PC1**, **PC2**, **PC3** y **PC4** que pertenecen a las redes 3FFE:8070:1024:**1**::/64, 3FFE:8070:1024:**2**::/64, 3FFE:8070:1024:**3**::/64 y 3FFE:8070:1024:**4**::/64 respectivamente, fijan su dirección IPv6 por el proceso de *autoconfiguración sin control de estado*, a partir de su Identificador de Interfaz y del prefijo de red anunciado por su enrutador.

4.3.7 **Procedimiento Fase 1**

Este procedimiento se basa en cada una de las tareas presentadas anteriormente para la configuración básica de OSPFv6 en un equipo enrutador IPv6 utilizando *zebra*.

- En los enrutadores **A1**, **A2**, **A3** y **A4**, habilite OSPFv6 y configure los atributos OSPFv6.

- Verifique la configuración actual de OSPFv6: ***ospf6d-ipv6****# show running-config*

- Ejecute en los enrutadores **A1**, **A2**, **A3** y **A4** el comando ***ospf6d-ipv6****# show ipv6 ospf6 route*, para examinar la tabla de enrutamiento OSPFv6 de estos enrutadores.

P.4.3.1 Analice la tabla de enrutamiento en cada uno de los enrutadores y explique la presencia o no de las redes 3FFE:8070:1024:**1**::/64, 3FFE:8070:1024:**3**::/64 y 3FFE:8070:1024:**4**::/64 en los enrutadores respectivos.

- En el enrutador A1 ejecute el comando # *ifconfig eth0 down* y observe nuevamente las tablas de enrutamiento en **A2**, **A3** y **A4**.

P.4.3.2 ¿Observó cambios en la tabla de enrutamiento de **A2**? Justifique su respuesta.

- En **A1** desconecte eth1 durante de 40 segundos y observe la tabla de enrutamiento en **A2**, luego conecte eth1 y vuelva a observar la tabla de enrutamiento.

P.4.3.3 ¿Qué cambios observo en la tabla de enrutamiento de **A2**? ¿Cuál es su concepto sobre la convergencia de OSPFv6?

- En **A1** desconecte eth0 y observe la tabla de enrutamiento en **A2** nuevamente conecte eth0 y observe la tabla de enrutamiento de **A2**.

P.4.3.4 ¿Qué puede concluir al respecto?

- Utilice el analizador de protocolos de red Ethereal[72] para identificar los tipos y el formato de los mensajes OSPFv6 que envía y recibe el enrutador **A2**. Para esto ejecute desde la consola de Linux de **A2** el comando # *ethereal*.

P.4.3.5 Describa los tipos y el formato de los mensajes OSPFv6

P.4.3.6 Describa el proceso de encapsulación de OSPFv6 en IPv6.

Fase 2: Múltiples Caminos OSPFv6

La Figura 56, describe la topología de la red piloto UniCauca IPv6 que se debe implementar en la Fase 2.

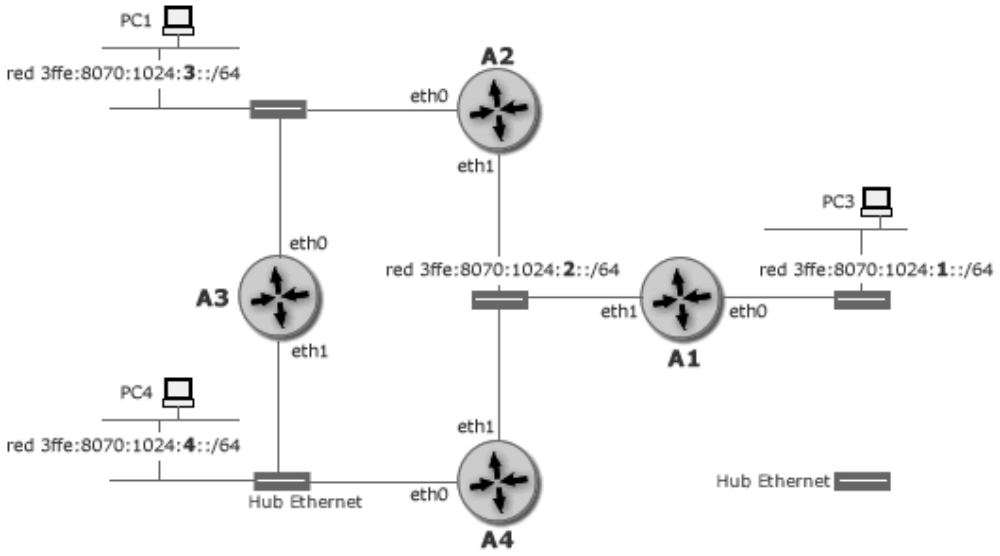


Figura 56 - Topología de la Fase 2

[72] Referirse a la sección 3.3.1: Instalación de Red Hat Linux 9 y de las aplicaciones utilizadas en el Laboratorio Internet IPv6. Se recomienda filtrar el protocolo OSPFv6. Información detallada acerca del funcionamiento de *ethereal* se encuentra en: http://www.ethereal.com/docs/

La Tabla 17, describe las direcciones IPv6 del tipo Unicast Global Agregable asignadas a las interfaces Ethernet de los enrutadores del SA UniCauca IPv6.

Enrutador	**A1**	**A2**	**A3**	**A4**
[eth0]	3FFE:8070:1024:1::**1**/64	3FFE:8070:1024:3::**2**/64	3FFE:8070:1024:3::**3**/64	3FFE:8070:1024:4::**4**/64
[eth1]	3FFE:8070:1024:2::**1**/64	3FFE:8070:1024:2::**2**/64	3FFE:8070:1024:4::**3**/64	3FFE:8070:1024:2::**4**/64

Tabla 17 - Direcciones IPv6 de los Enrutadores

Los hosts **PC1**, **PC3** y **PC4** que pertenecen a las redes 3FFE:8070:1024:**1**::/64, 3FFE:8070:1024:**3**::/64 y 3FFE:8070:1024:**4**::/64 respectivamente, fijan su dirección IPv6 por el proceso de *autoconfiguración sin control de estado*, a partir de su Identificador de Interfaz y del prefijo de red anunciado por su enrutador.

4.3.8 **Procedimiento Fase 2**

- Ejecute el comando ***ospf6d-ipv6****# show running-config* en todos los enrutadores y observe los costos configurados por defecto.

- Ejecute el comando ***ospf6d-ipv6****# show ipv6 ospf6 route* en los cuatro enrutadores.

P.4.3.7 Identifique el número de rutas (route), el camino (path) y las rutas redundantes en cada una de las tablas de enrutamiento OSPF de los enrutadores **A1**, **A2**, **A3** y **A4**, y justifique la elección que realiza OSPF escogiendo la ruta libre más corta a un destino específico.

- En las interfaces eth0 de los enrutadores **A2** y **A4** configure el atributo de costo ospf según la Tabla 18 para determinar diferentes criterios en el enrutamiento de la topología de red, luego observe la tabla de enrutamiento en **A1** y complete la columna “Próximo salto en A1”.

Redes	**Costo eth0 A2**	**Costo eth0 A4**	**Próximo salto en A1**
3FFE:8070:1024:**3**::/64	10	1	
3FFE:8070:1024:**4**::/64			
3FFE:8070:1024:**3**::/64	1	10	
3FFE:8070:1024:**4**::/64			

Tabla 18 - Configuración del atributo de costo OSPFv6

- Desconecte eth0 en **A2** y observe la tabla de enrutamiento en **A1**, repita el mismo proceso cambiando los valores de costo, asignando 10 para **A2** y 1 para **A4.**

P.4.3.8 Justifique los cambios.

- En **A1** ejecute traceroute6 a la dirección IPv6 de la interfaz eth0 **A3**, luego cambie los costos de **A2** =1 y **A4**=10 y ejecute traceroute6 para eth1 en **A1.**

P.4.3.9 ¿Son consecuentes los resultados de la tabla con los obtenidos en el traceroute6 anterior?

Fase 3: Coexistencia entre OSPFv6 como IGP y BGP4+ como EGP en el SA UniCauca IPv6

La Figura 57, describe la topología de la red piloto UniCauca IPv6 que se debe implementar en la Fase 3. Esta fase retoma la configuración OSPFv6 de la topología inmediatamente anterior y facilita la salida a 6bone del SA UniCauca IPv6 al permitir coexistir OSPFv6 y BGP4+.

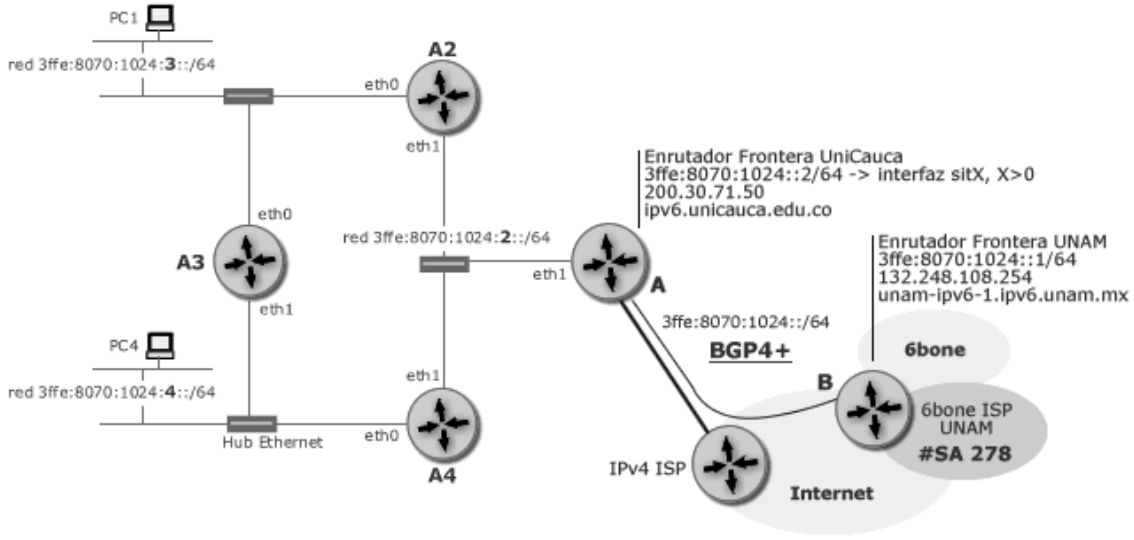


Figura 57 - Topología de la Fase 3

4.3.9 **Procedimiento Fase 3**

Este procedimiento se basa en cada una de las tareas presentadas anteriormente para la configuración básica de OSPFv6 y BGP4+ en un equipo enrutador IPv6 utilizando *zebra*.

- En el enrutador **A**, configure el proceso de enrutamiento BGP4+ utilizando el #SA 64515.

- En el enrutador **A**, configure iguales o vecinos BGP4+, para establecer al enrutador **B** 3FFE:8070:1024::1/64 como igual o vecino BGP4+.

- En el enrutador **A**, anuncie rutas a través de BGP4+ para publicar el espacio de direcciones IPv6 3FFE:8070:1024::/48 al SA UNAM IPv6.

- Verifique la configuración actual de BGP4+: ***bgpd-ipv6****# show running-config*

- Verifique la configuración actual de OSPF6d: ***ospf6d-ipv6****# show running-config*

- En el enrutador **A**, redistribuya BGP4+ en OSPFv6 ejecutando el siguiente comando: **ospf6d-ipv6(config-router)#** *redistribute bgp*

- Ejecute en los enrutadores **A2**, **A3**, **A4** el comando: ***ospf6d-ipv6****> show ipv6 ospf6d*, para examinar la tabla de enrutamiento OSPFv6 de estos enrutadores.

- Observe la tabla de enrutamiento en **A** tanto de BGP4+ como OSPFv6

P.4.3.10 ¿Cuál es su concepto sobre la gateway que aparece en las tablas de enrutamiento tanto en **A2** como **A3**?

- Verifique lo anterior ejecutando el comando *traceroute6* desde el enrutador **A3** hacia el extremo *remoto* del túnel, enrutador principal del sitio UNAM IPv6 y a un equipo cualquiera en 6bone, por ejemplo 3FFE:B00:C18:1::10. Para esto ejecute los comandos: *#traceroute6 3ffe:8070:1024::1* y *# traceroute6 3ffe:b00:c18:1::10*

4.3.10 **Análisis de Resultados**

4.3.11 **Conclusiones**

5 Laboratorio Internet IPv6: Seguridad

Arquitectura de Seguridad para IP **RFC2401**
Cabecera de Autenticación IP **RFC2402**
Cifrado de datos en IP (ESP) **RFC2406**
Asociaciones de Seguridad y Protocolo de Gestión de Claves en Internet (ISAKMP) **RFC2408**

5.1 Motivación

Esta práctica explica como configurar las características de seguridad en el núcleo del protocolo IPv6, al comprender la arquitectura y los servicios de seguridad que provee el protocolo IPSec.

5.2 Objetivos

- Habilitar y configurar IPSec: AH y/o ESP en modo Transporte.
- Habilitar y configurar IPSec: AH y/o ESP en modo Túnel.

5.3 Marco teórico

El *Internet Protocol Security Option*, opción de seguridad para el protocolo de Internet (IPSec), es un conjunto de estándares del IETF que incorpora servicios de seguridad en el núcleo del protocolo IPv6[73] al garantizar confidencialidad, control de acceso, protección antirepetición, integridad y autenticación en el nivel IP, de tal forma que su funcionamiento es completamente transparente a los protocolos de capas superiores.

- Confidencialidad: Implica que la información sea accesible únicamente por las entidades, sistemas o personas autorizadas.

- Control de acceso: Establece la forma en que el recurso esté disponible cuando es requerido.

- Protección antirepetición: Es la garantía de transmisión y recepeción de la información, busca proteger al emisor de que el receptor niegue haber recibido el mensaje, y proteger al receptor de que el transmisor niegue haber enviado el mensaje.

- Integridad: Implica que los datos no han sido modificados o corrompidos de manera alguna desde su transmisión hasta su recepción.

[73] IPSec está en capacidad de integrarse a IPv4 y se incluye por defecto en IPv6.

- Autenticación: Define mecanismos para garantizar la procedencia de la información, ya sea a nivel de usuario o de equipo de cómputo.

5.3.1 **Arquitectura de IPSec**

La arquitectura de IPSec está compuesta por 3 elementos básicos que trabajan en conjunto:

Figura 58 - Componentes de la Arquitectura IPSec

5.3.1.1 **Protocolos de Seguridad**

IPSec es un conjunto de protocolos que proporcionan servicios de seguridad para proteger tráfico IP. Estos servicios de seguridad trabajan gracias a dos protocolos, el Authentication Header (AH) y el Encapsulating Security Payload (ESP), además de protocolos y mecanismos para la gestión de llaves cifradas tales como IKE (Internet Key Exchange Protocol). De esta manera, el éxito de una implementación IPSec depende en gran medida de una adecuada escogencia del protocolo de seguridad y de la forma como se intercambian las llaves cifradas.

El protocolo AH garantiza la autenticidad e integridad de los paquetes IPv6, proporcionando un mecanismo al receptor de los paquetes para autenticar el origen de los datos y para verificar que estos datos no hayan sido alterados en la transmisión. Sin embargo, AH no garantiza confidencialidad, es decir, los datos transmitidos pueden ser vistos por terceros.

El protocolo ESP sí garantiza confidencialidad para el tráfico IPv6 y adicionalmente ofrece servicios de integridad y autenticidad en el origen de los datos incorporando un mecanismo similar al de AH, pero solo uno de estos servicios puede ser proporcionado por ESP al mismo tiempo.

El protocolo IKE permite a dos nodos negociar dinámicamente sus llaves de cifrado y todos los parámetros necesarios para establecer una conexión AH o ESP.

5.3.1.2 **Asociaciones de Seguridad (AS)**

Una AS es un canal de comunicación unidireccional[74] que conecta dos entidades IPSec, las cuales desean comunicarse de forma segura utilizando servicios de seguridad acordados previamente. El conjunto de servicios de seguridad ofrecidos por una AS depende de los protocolos de seguridad y del modo en el cual ellos operan, definidos por la AS misma.

La Figura 59, muestra los 2 modos en los cuales un protocolo de seguridad puede operar: *Transporte* y *Túnel*; la diferencia radica en la manera como cada uno de ellos altera el paquete IPv6 original. En modo Transporte la cabecera básica IPv6 se mantiene intacta y una cabecera de Seguridad es colocada entre la cabecera básica y su carga útil, así se protegen los protocolos de capas superiores tales como TCP y UDP. En modo Túnel, el paquete IPv6 original se convierte en la carga útil de un nuevo paquete IPv6, esto le permite al paquete IPv6 original, ocultar su cabecera para que sea cifrada, considerando que el paquete IPv6 externo sirve de guía a los datos a través de la red.

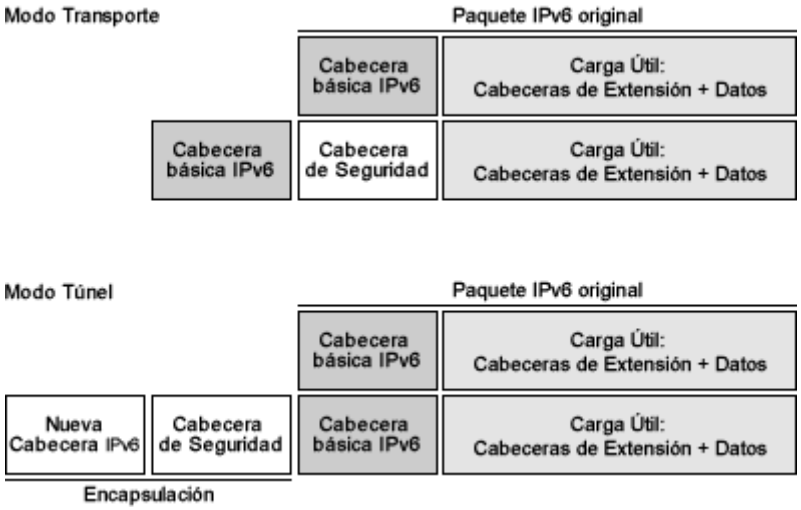


Figura 59 - Estructura de un paquete IPv6 en modo Transporte y Túnel

Una AS se identifica por tres parámetros:

- Dirección IPv6 Destino[75]: Dirección Unicast que identifica el destino de la AS.
- Identificador de protocolo de Seguridad: Número que identifica el tipo de protocolo de Seguridad que se utiliza, 51 para AH o 50 para ESP.
- Índice de parámetro de Seguridad (**SPI**): Indica cuales son los parámetros de Seguridad específicos a la AS que se está utilizando. Es el mecanismo concebido para que en una

[74] Una AS se define entre dos hosts o gateways para datos enviados en una sola dirección, de aquí que, si 2 dispositivos necesitan intercambiar información en ambas direcciones usando IPSec, requerirán de dos AS's, una para cada sentido.
[75] Direcciones IPv6 de origen no se usan para definir AS debido a que una AS se define en una sola dirección, en consecuencia, si dos dispositivos necesitan intercambiar información en ambas direcciones usando IPSec, requieren de dos AS, una en cada sentido.

comunicación segura, el origen identifique cual AS utilizar para asegurar un paquete por enviar, y el destino identifique cual AS utilizar para verificar la seguridad del paquete recibido. El SPI se incluye en los encabezados AH y ESP, y el destino utiliza el conjunto - *SPI, Destino, Protocolo* - para identificar de forma única la AS.

5.3.1.3 **Bases de datos de Seguridad**

IPSec trabaja con 2 bases de datos de seguridad:

- *Security Association Database*, base de datos de Asociaciones de Seguridad (SAD)
- *Security Policy Database*, base de datos de Políticas de Seguridad (SPD)

Base de datos de Asociaciones de Seguridad (SAD)

La base de datos de Asociaciones de Seguridad almacena todos los parámetros concernientes a una AS, cada AS tiene una entrada en la SAD donde se especifican todos los parámetros necesarios para que IPSec realice el procesamiento de los paquetes IPv6 que son administrados por esa AS. Entre los parámetros que se encuentran en una SAD se tienen:

- El Índice de parámetro de seguridad (SPI)
- El protocolo a ser usado por la AS (ESP o AH)
- El modo en el cual el protocolo es operado (transporte o túnel)
- Un contador numérico secuencial
- La dirección IPv6 Origen y Destino de la AS
- El algoritmo de autenticación y la llave de autenticación utilizadas
- El algoritmo de cifrado y su llave
- El tiempo de vida de las llaves de autenticación y de cifrado
- El tiempo de vida de la AS

Base de datos de Políticas de Seguridad (SPD)

Una base de datos de Políticas de Seguridad es una lista ordenada de políticas de seguridad a ser aplicadas a los paquetes IPv6. Dichas políticas son en general reglas que especifican como los paquetes IPv6 deben ser procesados.

Para todo paquete entrante o saliente existen 3 acciones posibles:

- Aplicar IPSec: Consiste en cifrar y/o autenticar un paquete de acuerdo con las normas establecidas en la propia entrada de la base de datos. Cuando esto se hace se comprueba

si la entrada correspondiente está vinculada a una de la SAD, de lo contrario se crea una nueva y se procede al envío de la información.

- No aplicar IPSec: Consiste en realizar un encaminamiento normal del paquete IPv6, o tratarlo como lo haría un sistema que no implementa IPSec.

- Descartar el paquete: Significa inhibir todo tipo de acción, similar a lo que haría un firewall.

La SPD trata al tráfico saliente y entrante de manera separada, esto es, que se deben aplicar políticas de seguridad distintivas de entrada y de salida por cada interfaz de red. Cuando un paquete IPv6 llega a una interfaz de red, IPSec primero busca en la SAD la apropiada AS, cuando la encuentra, el sistema inicia los procesos SAD y SPD. Después del procesamiento SPD, el sistema reenvía el paquete al siguiente salto o le aplica procedimientos adicionales tales como las reglas de algún firewall.

El procesamiento PSD se realiza primero en paquetes salientes. Si la entrada SPD que concuerda especifica que un procesamiento IPSec es necesario, la SAD es consultada para determinar si una SA ha sido previamente establecida, en caso contrario se negocia una nueva SA para el paquete.

Dado que los procesos SPD son realizados tanto para los paquetes IP salientes como entrantes, este procedimiento puede alterar negativamente el desempeño de un dispositivo IPSec. De hecho, el procesamiento SPD es el cuello de botella más representativo en una implementación IPSec.

Nota: La SPD es mantenida por el administrador de la entidad IPSec.

5.3.2 **Authentication Header (AH)**

El protocolo AH es usado para propósitos de autenticación de la carga útil IPv6 a nivel de paquete por paquete, esto es autenticación de la integridad de los datos y del origen de los mismos.

Como el término autenticación lo indica, el protocolo AH se asegura que los datos entregados dentro del paquete IPv6 son auténticos, es decir, define mecanismos para garantizar la procedencia de la información. AH también provee un mecanismo de protección opcional antirepetición de paquetes IPv6. Sin embargo, AH no protege la confidencialidad de los datos, es decir, no recurre a ningún tipo de cifrado de los mismos.

El protocolo AH define como un paquete IPv6 sin protección es convertido en uno nuevo que contiene información adicional y que brinda autenticación. El elemento fundamental usado por AH es una cabecera de Autenticación como se muestra en la Figura 60. El nuevo paquete IPv6 se forma insertando la cabecera de Autenticación, bien sea, después de la nueva cabecera IP o después de la cabecera IP original modificada según sea el modo en el cual trabaje la AS: Transporte o Túnel.

Cuando la cabecera de Autenticación es insertada, la cabecera IPv6 que la precede deberá indicar que la Siguiente Cabecera que se encuentra es la cabecera de Autenticación y no la carga útil del paquete original. La cabecera IPv6 realiza esta acción colocando en el campo Siguiente Cabecera el valor 51 (valor de protocolo para AH).

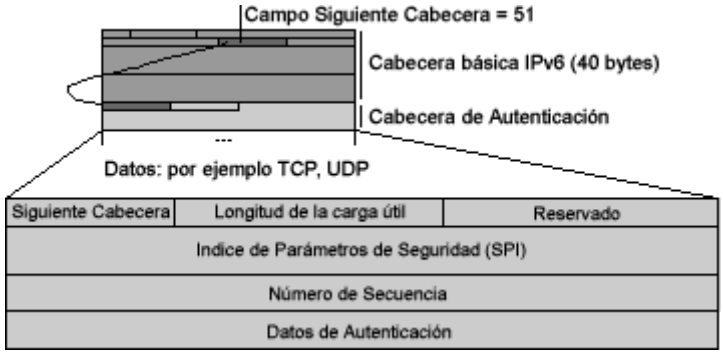


Figura 60 - Formato de AH

La siguiente lista describe cada campo:

- Siguiente Cabecera: Este campo identifica el tipo de cabecera que sigue inmediatamente a la cabecera de Extensión AH.

- Longitud de la carga útil: Este campo identifica la longitud la cabecera de Extensión AH.

- Reservado: No utilizado, este campo se fija a 0.

- Índice de Parámetros de Seguridad: Este campo es un número arbitrario de 32bits, utilizado junto con la dirección IPv6 de destino y el tipo de protocolo IPSec (en este caso AH) para identificar la AS para este paquete IPv6.

- Número de Secuencia: Este campo mantiene un incremento monotónico de la secuencia de paquetes IPSec. Comienza en 0 cuando la AS es establecida y se incrementa por cada paquete IPv6 saliente que utiliza esta AS. Este campo se emplea como un mecanismo de protección antirepetición.

- Datos de Autenticación: Este en un campo de longitud variable que contiene el *Integrity Check Value*, Valor de chequeo de integridad (ICV) para este paquete IPv6. El ICV es calculado con el algoritmo seleccionado por la AS y es utilizado por el receptor para verificar la integridad del paquete IPv6 entrante. Los algoritmos por defecto requeridos por AH para trabajar son HMAC con MD5 y SHA-1. El ICV es un valor hash[76] computado sobre todos los campos que la autenticación incluye. La llave secreta es negociada durante el establecimiento de la AS. La autenticación de un paquete recibido es verificada cuando el receptor calcula el valor hash y lo compara con el ICV del paquete entrante. Si el paquete IPv6 no es autenticando exitosamente entonces es descartado.

5.3.2.1 **Modo Transporte para AH**

En modo transporte, la cabecera básica del paquete IPv6 original es conservada como la cabecera del nuevo paquete IPv6, y la cabecera de Autenticación es insertada entre la cabecera IPv6 y la carga útil original, como se muestra en la Figura 61. Finalmente el ICV es calculado sobre la totalidad del nuevo paquete IPv6.

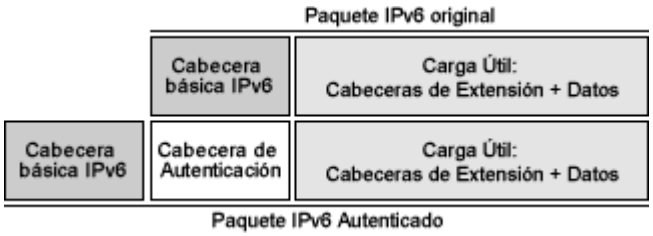


Figura 61 - Modo de Transporte AH

La ventaja del modo Transporte es que solo adiciona unos pocos bytes extra al paquete IPv6 original. Sin embargo, por conservarse la cabecera IPv6 original como la misma cabecera del nuevo paquete IPv6, solamente puede ser utilizado por hosts finales, esta es una limitación grande cuando los dispositivos que están administrados por esta AS actúan como gateways de otros hosts que se encuentran detrás de ellos.

5.3.2.2 **Modo Túnel para AH**

En modo Túnel, una nueva cabecera es creada para el nuevo paquete IPv6 y la cabecera de Autenticación es insertada entre las cabeceras nueva y original, tal como se muestra en la Figura 62. El paquete IPv6 original permanece intacto y es encapsulado dentro del nuevo paquete IPv6. De esta manera, la autenticación se aplica sobre el paquete IPv6 original entero.

[76] Algoritmo de cifrado que convierte un mensaje de cualquier longitud en una sola cadena de dígitos.

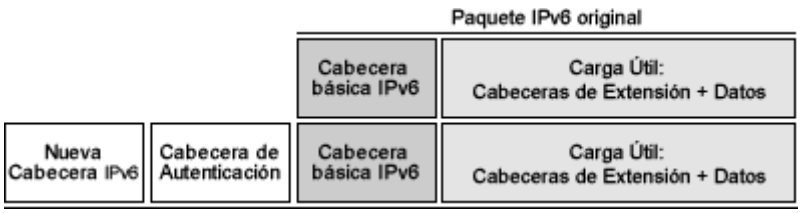


Figura 62 - Modo Túnel AH

La cabecera IPv6 original permanece completamente inalterada y contiene las direcciones IPv6 tanto de destino como origen de los dispositivos que emiten y reciben el tráfico IP original. La nueva cabecera IPv6 contiene la dirección IPv6 origen y destino de los dispositivos IPSec entre los cuales viaja el nuevo paquete. De esta manera el modo Túnel puede ser usado si los puntos finales de la AS son un host o una gateway de seguridad.

A diferencia del modo Transporte, el modo Túnel tiene como desventaja adicionar más bytes extra por lo cual el throughput efectivo del enlace disminuye al igual que el desempeño de los dispositivos se torna más lento por el doble procesamiento de cabecera que se necesita.

5.3.3 **Encapsulating Security Payload (ESP)**

El protocolo ESP provee autenticación, confidencialidad de los datos por medio de cifrado y una protección opcional antirepetición para los paquetes IPv6. La autenticación y el cifrado son también opcionales, pero al menos una de ellas debe ser empleada; de lo contrario, este protocolo carecería de fundamento.

La confidencialidad es lograda por medio de técnicas de cifrado. Los algoritmos de cifrado empleados para los paquetes IPv6 son definidos por la AS sobre la cual los paquetes son enviados. El algoritmo de cifrado Null en el cual el cifrado no es aplicado, es también un algoritmo válido en este protocolo. En este caso, ESP solamente presta el servicio de autenticación para el tráfico.

Al igual que con AH varios campos adicionales son insertados en el paquete IPv6 para que presten los servicios mencionados anteriormente. La Figura 63 muestra la conformación de un paquete IPv6 después que se ha procesado con el protocolo ESP.

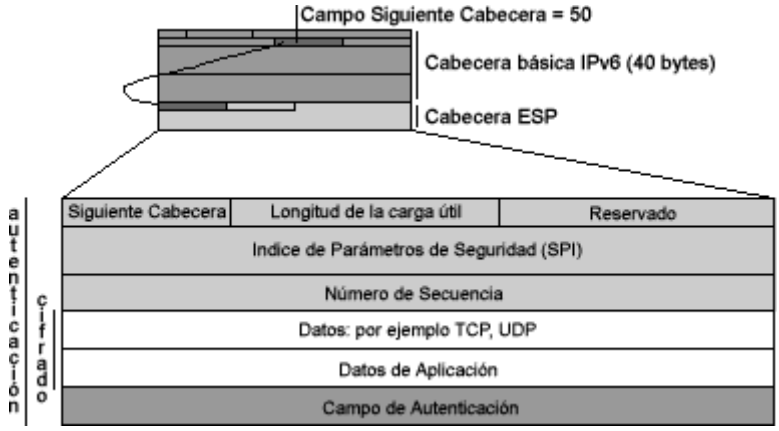


Figura 63 - Nuevo paquete IP procesado con ESP

La cabecera ESP se encuentra después de la nueva cabecera IPv6 o después de la cabecera IPv6 original modificada, esto dependiendo del modo. Al igual que en el protocolo AH, los campos SPI, Número de Secuencia, Siguiente Cabecera y Autenticación, se encuentran definidos a lo largo del nuevo paquete IP.

Existe un problema cuando la longitud del nuevo paquete IPv6, debido a la adición de una cabecera ESP y de unos campos de relleno y de autenticación, resulta ser mas grande que el tamaño máximo definido para el paquete (MTU). Cuando esto pasa, los paquetes IPv6 son fragmentados por el origen. Debido a que el procesamiento ESP debe ser aplicado únicamente a paquetes IPv6 enteros y no fragmentados, si un paquete IP entrante ha llegado fragmentado, el gateway de seguridad que lo recibe debe reensamblar los fragmentos para formar de nuevo el paquete IPv6 antes de que sean procesados por ESP.

5.3.3.1 **Modo Transporte para ESP**

En el modo transporte, la cabecera ESP es insertada entre la cabecera básica IPv6 y la carga útil original. Dado que la cabecera IPv6 original sigue sin alteraciones, el modo de transporte para el protocolo ESP, al igual que en AH, solamente puede ser usado entre hosts.

El modo de transporte es el más usado cuando no es necesario ocultar o autenticar las direcciones IPv6 tanto del origen como del destino.

En la Figura 64 se detalla la conformación del nuevo paquete IPv6 usando ESP en modo transporte, además se muestra la parte del paquete que puede ser cifrada y la parte del paquete que puede ser autenticada.

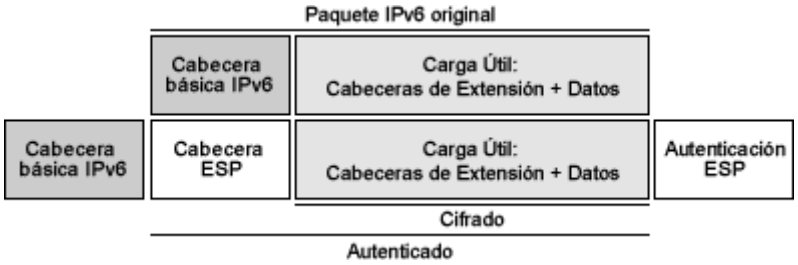


Figura 64 - Modo Transporte ESP

5.3.3.2 **Modo Túnel para ESP**

En modo túnel, el paquete IP original enteramente es encapsulado dentro de un nuevo paquete IPv6. En la Figura 65 se muestra como la nueva cabecera IPv6 y la cabecera ESP son puestas al comienzo del paquete IP original. Si el túnel se encuentra establecido entre hosts, las direcciones IP origen y destino, en la nueva cabecera IP pueden ser las mismas que en la cabecera original. Si el túnel se encuentra establecido entre dos gateways de seguridad, las direcciones en la nueva cabecera IPv6 serán las direcciones de los gateways. Ejecutando ESP en modo túnel entre gateways de seguridad se puede lograr tanto confidencialidad como autenticación del tráfico en tránsito entre los 2 gateways.

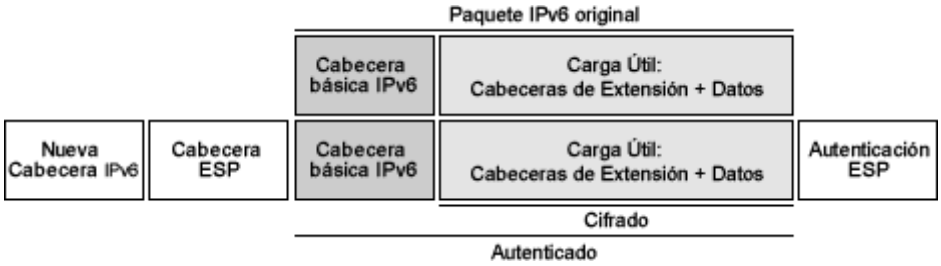


Figura 65 - Modo Túnel ESP

5.3.4 **Internet Key Exchange (IKE)**

Los protocolos ESP y AH no explican cómo las asociaciones de seguridad son negociadas, simplemente se refiere a cómo lo servicios de seguridad son aplicados a cada paquete IPv6 de acuerdo con lo que le indica la AS. Las AS's pueden ser configuradas manualmente por el administrador del sistema o pueden ser negociadas dinámicamente por medio de un protocolo de manejo de llaves tal como IKE.

Este último tipo de negociación es muy importante debido a que en una comunicación de datos es imposible saber cuando una nueva AS se tiene que establecer y más cuando los datos a asegurar provienen del exterior del sistema. La otra razón importante para usar negociación dinámicas de AS, es que por motivos de seguridad las AS no pueden tener un tiempo de vida muy largo, dado que se expone a que algún atacante rompa los códigos de seguridad. Para

eliminar este riesgo, las AS se renegocian periódicamente regenerando así todo el material asociado a las llaves mismas.

IKE está basado en el *Internet Security Association And Key Management Protocol*, Protocolo de gestión de llaves y de Asociaciones de Seguridad en Internet (ISAKMP), el cual implementa elementos de los métodos de intercambio de llaves Oakley y SKEME (Secure Key Exchange Mechanism).

ISAKMP define un conjunto de procedimientos por medio de los cuales se asegura el canal de comunicación entre dos puntos. En otras palabras, es el protocolo encargado de preparar el canal de transporte seguro que luego será usado por las AS para ser negociadas.

Hay 2 fases en la negociación ISAKMP de una AS. La primera fase es la negociación entre los dos nodos ISAKMP. En esta fase dos nodos se ponen de acuerdo en la forma de proteger las comunicaciones que se establecerán luego entre ellos, se puede decir que en esta fase se crea una AS ISAKMP. Es muy importante no confundir una ISAKMP AS con las AS propias de IPSec tratadas anteriormente. Una ISAKMP AS es bidireccional y no trabaja sobre el tráfico IPSec. En la segunda fase, las AS propias de IPSec son negociadas entre los dos nodos ISAKMP. Dado que el canal se ha asegurado en la primera fase, las negociaciones dentro de esta segunda fase se desarrollan de una manera mas sencilla. Una misma AS ISAKMP puede ser usada para negociar muchas AS IPSec reduciendo el número de negociaciones. Lo anterior sucede en aplicaciones IPSec LAN-a-LAN donde un gateway de seguridad actúa como un nodo ISAKMP en nombre de los hosts que él protege.

5.3.4.1 **Generación de Llaves en IKE**

Las llaves son generadas una vez los parámetros necesarios se encuentran en los nodos ISAKMP. El algoritmo Diffie-Hellman juega un papel vital en la generación de llaves en IKE. Éste permite que dos partes generen una llave secreta a partir de parámetros públicos, de tal manera que una tercera entidad que tenga la intención de obtener la llave secreta “escuchando” la comunicación de los dos nodos involucrados sea incapaz de hacerlo.

Sin embargo, el protocolo Diffie-Hellman es vulnerable a ataques en los cuales alguien intercepta los mensajes que se intercambian y suplanta uno de los nodos. Por lo anterior, en el intercambio IKE, las dos partes involucradas deben ser autenticadas.

La llave secreta Diffie-Hellman generada en la fase 1 es llamada la ISAKMP master key y la llave secreta Diffie-Hellman generada en la fase 2 es llamada la user master key.

Oakley es un protocolo que se usa para determinar llaves y que se basa en el esquema Diffie-Hellman para intercambiar llaves secretas de una manera segura entre dos partes que se han autenticado previamente.

5.4 Desarrollo Técnico

5.4.1 Implementación de IPSec en RH Linux

Para implementar IPSec en Red Hat Linux se utilizarán las fuentes del kernel USAGI[77] IPSec, las cuales proveen una implementación de los protocolos AH y ESP para IPv6 basados en un kernel 2.4.21. Así USAGI IPSec convierte un equipo Linux en una gateway segura IPSec, permitiendo implementar topologías HOST-a-HOST / LAN-a-LAN[78] y soportando los modos Transporte y Túnel para AH y ESP.

Instalación de USAGI STABLE Release 5[79]

- Obtener y descomprimir la versión estable de las fuentes del kernel USAGI STABLE Release 5: usagi-linux24-stable-20040104.tar.bz2 desde ftp://ftp.linux-ipv6.org/pub/usagi/stable/kit/ y ejecutar el comando: *#tar -jxvf **ruta-fuentes-USAGI**/usagi-linux24-stable-20040104.tar.bz2 –C /usr/src*

- Ingrese al directorio de fuentes del kernel: *#cd /usr/src/usagi/kernel/linux-**version***

- Configure el kernel para el soporte de IPSec IPv6:
  #make xconfig ó make menuconfig

Grupo	Opción	Selección
Code maturuty level options	Prompt for development and/or incomplete code/drivers	[*]
Loadable module support	Enable loadable module support	[*]
	Set version information on all module symbols	[*]
	Kernel module loader	[*]
Cryptografhy suport (CryptoAPI)	CryptoAPI support	[*]
	Cipher Algorithms	[*]
	--- 128 bit blocksize	
	AES (aka Rijndael) cipher	[*]
	--- 64 bit blocksize	
	3DES cipher	[*]
	--- Deprecated	
	NULL cipher (NO CRYPTO)	[*]

[77] Proyecto USAGI: El cual es un esfuerzo de empresas japonesas para implementar todo el stack del protocolo IPv6 en Linux - http://www.linux-ipv6.org
[78] Una sesión segura IPSec IPv6 en un escenario HOST-a-LAN, aún no alcanza una desarrollo estable, y esta es la razón por la que no se incluye en este Laboratorio.
[79] Se recomienda ejecutar los comandos *#/sbin/lspci* y *#cat /proc/cpuinfo*, para obtener información detallada acerca de las características HW del PC. En todo caso es adecuado utilzar un Kernel-HOWTO: www.tldp.org/HOWTO/Kernel-HOWTO.html

	DES cipher (DEPRECATED)	[*]
	Digest Algorithms	[*]
	MD5 digest	[*]
	SHA1 digest	[*]
Networking options	Packet socket	[*]
	Unix domain sockets	[*]
	TCP/IP networking	[*]
	IP: tunneling	[*/M]
	The IPsec protocol (EXPERIMENTAL)	[*]
	IPsec: Ipsec Debug messages	
	IPsec: Ipsec debugging off by default	
	The IPv6 protocol (EXPERIMENTAL)	[*/M]
	....	
	IPv6: IP Security Support (EXPERIMENTAL)	[*]
	IPv6: IPv6 over IPv6 Tunneling (EXPERIMENTAL)	<M>

Tabla 19 - Opciones del kernel específicas para IPSec IPv6

Compilación e Instalación

- Compile el nuevo kernel: #*make dep*; #*make clean*; #*make* {*zImage* | *bzImage*};

- Compile los módulos: #*make modules*; #*make modules_install*

- Cree una imagen inicial RAMDisk: #*mkinitrd /boot/initrd-**version**.img version*

- Remueva el enlace simbólico al mapa del kernel, System.map: #*rm /boot/System.map*

- Copie y renombre el nuevo kernel y el System.map:
  #*cp arch/i386/boot/*{*zImage* | *bzImage*} */boot/vmlinuz-**version***
  #*cp System.map /boot/System.map-**version***

- Cree un enlace simbólico al *System.map-**version***: #*ln -s /boot/System.map-**version** /boot/System.map*

- Edite */etc/lilo.conf* o */etc/grub.conf* y añadir la entrada al nuevo kernel

- Reinicie el sistema y seleccionar el nuevo kernel: #*reboot*

Arquitectura de USAGI IPSec

La arquitectura de USAGI IPSec provee 2 componentes fundamentales para la operación de IPSec en Linux:

- ***pfkey*** - que es una utilidad para crear, configurar, eliminar y desplegar de forma manual asociaciones y políticas de seguridad IPSec.

- ***pluto*** - que es una herramienta que permite implementar el protocolo IKE.

En consecuencia IPSec puede ser configurado manualmente utilizando el comando *pfkey* o puede ser configurado dinámicamente a través del demonio IKE que facilita *pluto*.

Instalación del comando *pfkey*

- Ingrese al directorio pfkey: *#cd /usr/src/usagi/usagi/pfkey/* y ejecute: *#./configure; make; make install*

Instalación del *pluto*

- Ingrese al directorio pluto: *#cd /usr/src/usagi/usagi/pluto/* y ejecute: *#./configure; make; make install*

Nota: Ahora las herramientas y utilidades IPSec a utilizar se encuentran en: */usr/local/v6/sbin/*, y los archivos de configuración en: */usr/local/v6/etc/*

Trabajando con USAGI IPSec

Lo primero que hace IPSec en el momento de establecer una sesión segura es autenticarla, para esto determina si el equipo remoto con quien se desea establecer la sesión es quien dice ser, y si además es un equipo válido. Luego se establece la sesión y las partes comienzan a "hablar" empleando el protocolo AH o ESP.

Durante el proceso de autenticación USAGI IPSec utiliza dos tipos de autenticación: *Pre Shared Keys* (llaves compartidas preacordadas) y Firmas digitales RSA.[80]

Pre Shared Keys

Este tipo de autenticación utiliza una única llave en texto plano la cual comparten todos los nodos que pertenecen a una red segura. Un nodo IPSec que recibe la conexión desde otro nodo IPSec, determina la validez de la conexión, si el nodo origen tiene la llave correcta.

Entre las ventajas de utilizar *pre shared keys* se tiene:

- Facilidad para configurar este tipo de autenticación.
- Autenticación disponible en la mayoría de implementaciones IPSec.

[80] Existes paquetes precompilados y parches para dar soporte a los Certificados X.509 que utiliza IPSec como otro tipo de Autenticación, pero éstos aún no son versiones estables en IPv6.

Una desventaja de este tipo de autenticación es la pobre gestión de éstas llaves.

Firmas digitales RSA

Las firmas digitales son firmas asincrónicas[81] que se basan en el manejo de una pareja de llaves. Cada llave puede cifrar información que solo la otra puede descifrar. La llave privada, únicamente es conocida por su propietario; la llave pública, se publica abiertamente, pero sigue asociada al propietario.

Las llaves se pueden usar de dos maneras diferentes: para garantizar confidencialidad al mensaje y para probar la autenticidad del emisor de un mensaje. En el primer caso, el emisor usa la llave pública del receptor para cifrar un mensaje, de manera que el mensaje continúe siendo confidencial hasta que sea decodificado por el receptor con la llave privada. En el segundo caso, el emisor cifra un mensaje usando la llave privada, una llave a la cual sólo tiene acceso él.

La llave pública del receptor asegura la confidencialidad, la llave privada del emisor verifica la identidad del mismo.

5.4.2 **Configurando los tipos de Autenticaciones**

USAGI IPSec implementa por el momento dos tipos de autenticación: *Pre Shared Keys* y Firmas digitales RSA, cada uno tiene su propia forma de configuración:

- Pre Shared Keys: Para configurar este tipo de autenticación es necesario poner en un solo lugar la llave única compartida. Este lugar es el archivo */usr/local/v6/etc/ipsec.secrets*, y debe contener lo siguiente: *<direcciones IPv6 que intervienen> : PSK "llave".* Ejemplo:

```
3FFE:8070:1024:A:B:C:D::1 3FFE:8070:1024:A:B:C:D::2 : PSK “Proyecto UniCauca IPv6”
```

- Firmas digitales RSA

Para configurar este tipo de autenticación se debe ejecutar el siguiente comando: *#ipsec rsasigkey --verbose nbits > /usr/local/v6/etc/ipsec.secrets*, donde *nbits* identifica el número de bits a utilizar para construir la llave.

Este comando generará un archivo nuevo con un contenido similar al siguiente:

[81] La firma digital se divide en 2: una llave privada y una llave pública.

```
# RSA 1024 bits ipv6.unicauca.edu.co Tue Jan 27 01:14:02 2004
# for signatures only, UNSAFE FOR ENCRYPTION
#pubkey=0sAQPY+BA6k+J7emr7+.....
#IN KEY 0x4200 4 1 AQPY+BA6k+J7em.....
Modulus: 0xd8f8103a93e27b7a6afbfa6c6b.....
PublicExponent: 0x03
# everything after this point is secret
PrivateExponent: 0x24295809c35069e9bc7f546.....
Prime1: 0xed2ef593c6.....
Prime2: 0xea2e8fe4f956.....
Exponent1: 0x9e1f4e.....
Exponent2: 0x9c1f0a98a6.....
Coefficient: 0x497825b73814334.....
```

5.5 Recursos

- 4 PC's con S.O Red Hat Linux 9
- Espacio de direcciones IPv6 (*bloque* pNLA /48): 3FFE:8070:1024::/48

5.6 Desarrollo de la Práctica

Esta práctica se divide en 2 fases:

- Fase 1: Habilitación de AH y/o ESP en modo Transporte
- Fase 2: Habilitación de AH y/o ESP en modo Túnel

Fase 1: Habilitación de AH y/o ESP en modo Transporte

La Figura 66, describe la topología que se debe implementar:

Figura 66 - Escenario de Prueba: Modo Transporte

La configuración de las interfaces de red Ethernet en los host a y b es la siguiente:

	host**a**	host**b**
Interfaz Ethernet [eth0]	3FFE:8070:1024:1::**1**/64	3FFE:8070:1024:1::**2**/64

5.7 Procedimiento Fase 1

- Configuración Manual: En el escenario propuesto se debe tener en cuenta lo siguiente:

Modo Transporte AH/ESP		
Algoritmo de autenticación AH	hmac-md5 llave: 0x0123456789abcdef0123456789abcdef	
Algoritmo de autenticación ESP	hmac-md5 llave: 0x0123456789abcdef0123456789abcdef	
Algoritmo de cifrado ESP	3des-cbc llave: 0xa7a36ebd91863edfba763fa7edcba64d89123ace6359eba7	
SPI	host **a** >> host **b** - AH	0x1234
	host **a** >> host **b** - ESP	0x5678
	host **b** >> host **a** - AH	0x9abc
	host **b** >> host **a** - ESP	0xdef0

Configuración de Asosiaciones y Políticas de Seguiridad: host **a** >> host **b**

- Asociación de Seguridad para AH: *#pfkey* -A sa -s 3FFE:8070:1024:1::**1** -d 3FFE:8070:1024:1::**2** -T ah -S 0x1234 --auth hmac-md5 --authkey 0x0123456789abcdef0123456789abcdef

- Asociación de Seguridad para ESP: *#pfkey* -A sa -s 3FFE:8070:1024:1::**1** -d 3FFE:8070:1024:1::**2** -T esp -S 0x5678 --auth hmac-md5 --authkey 0x0123456789abcdef0123456789abcdef --esp 3des-cbc --espkey 0xa7a36ebd91863edfba763fa7edcba64d89123ace6359eba7

- Política de Seguirdad para AH: *#pfkey* -A sp -s 3FFE:8070:1024:1::**1** -d 3FFE:8070:1024:1::**2** -T ah -S 0x1234

- Política de Seguridad para ESP: *#pfkey* -A sp -s 3FFE:8070:1024:1::**1** -d 3FFE:8070:1024:1::**2** -T esp -S 0x5678

Configuración de Asosiaciones y Políticas de Seguiridad: host **b** >> host **a**

- Asociación de Seguridad para AH: *#pfkey* -A sa -d 3FFE:8070:1024:1::**1** -s 3FFE:8070:1024:1::**2** -T ah -S 0x9abc --auth hmac-md5 --authkey 0x0123456789abcdef0123456789abcdef

- Asociación de Seguridad para ESP: # *pfkey* -A sa -d 3FFE:8070:1024:1::**1** -s 3FFE:8070:1024:1::**2** -T esp -S 0xdef0 --auth hmac-md5 --authkey 0x0123456789abcdef0123456789abcdef --esp 3des-cbc --espkey 0xa7a36ebd91863edfba763fa7edcba64d89123ace6359eba7

- Politica de Seguridad para AH: #*pfkey* -A sp -d 3FFE:8070:1024:1::**1** -s 3FFE:8070:1024:1::**2** -T ah -S 0x9abc

- Politica de Seguridad para ESP: #*pfkey* -A sp -d 3FFE:8070:1024:1::**1** -s 3FFE:8070:1024:1::**2** -T esp -S 0xdef0

- Ejecute los comandos #*pfkey -L* o #*ipsec-conf show*, para desplegar la información de las asociaciones y de las políticas de seguridad establecidas.

- Guarde la configuración de las asociaciones y de las políticas de seguridad establecidas utilizando el comando #*ipsec-conf save* <*nombre-del-archivo*>*.*

- Puede recuperar la configuración de las asociaciones y de las políticas de seguridad salvadas en el archivo <*nombre-del-archivo*> utilizando el comando: #*ipsec-conf restore* <*nombre-del-archivo*>

- En el host**a** envíe ping6 al host**b**.

- Utilice el analizador de protocolos de red Ethereal para identificar los tipos de mensajes que recibe el host**b**. Para esto ejecute desde la consola de Linux el comando # *ethereal* sobre la interfaz eth0*.*

P.5.1 Describa los campos de las cabeceras AH y ESP.

- Observe el campo *datos* del paquete UDP y tome nota de los valores en hexadecimal. Para esto guarde el archivo que genera *ethereal.*

P.5.2 Describa el proceso de encapsulación de IPSec para su transporte sobre IPv6.

- Ejecute el comando #*ipsec-conf reset* para eliminar la configuración de las asociaciones y políticas de seguridad.

- Utilizando ping6 envíe tráfico ICMP del host **a** al host **b**.

- Utilice el analizador de protocolos de red Ethereal para identificar los tipos de mensajes que recibe el host **b**. Para esto ejecute desde la consola de Linux el comando # *ethereal* sobre la interfaz eth0*.*

P.5.3 Describa el formato y los tipos de mensajes recibidos por el host **b**.

- Observe el campo *datos* del paquete UDP y tome nota de los valores en hexadecimal. Para esto guarde el archivo que genera *ethereal.*

P.5.4 Ahora que puede comparar el campo datos en los paquetes UDP recibidos, habilitando o deshabilitando una sesión IPSec entre los hosts, ¿Qué puede concluir al respecto?

- Configuración Automática - IKE

USAGI IPSec provee un archivo principal de configuración y es el *ipsec.conf*, que se encuentra en */usr/local/v6/etc/*. El archivo *ipsec.conf* tiene dos tipos de secciones, la sección "config" de configuraciones y la sección "conn" de conexiones. En este momento la única sección de configuración que acepta USAGI IPSec es la sección "**config setup**".

La sección "config setup" tiene toda la información que el software necesita al inicializarse. Este es un ejemplo de esta sección:

```
config setup

interfaces=%defaultroute
klipsdebug=none
plutodebug=none
#plutoload=%search
#plutostart=%search
uniqueids=yes
```

El valor más importante en este ejemplo es el del parámetro interfaces, el valor especial %defaultroute significa que la interfaz de red que *pluto* va a utilizar para establecer sesiones IPSec, es la que utiliza la ruta por defecto.

Los parámetros *klipsdebug* y *plutodebug*, se utilizan cuando hay problemas más complejos. En los archivos de logs de Linux se encuentra suficiente información para determinar dónde están los problemas más comunes.

Secciones "**conn**"

Las secciones "conn" se utilizan para decirle a *pluto* que tipo de sesión IPSec se va a establecer o aceptar. Los cambios básicos que definen cada pareja IPSec son:

```
af=inet6
type=transport/tunnel
auth=
authby=
```

```
left=
leftsubnet=
leftnexthop=
right=
rightsubnet=
rightnexthop=
```

- El campo *af=inet6* identifica la familia de direcciones IPv6.

- El campo *type* especifica los dos modos en los cuales IPSec puede operar: Transporte y Túnel.

- Los campos *auth* y *authby* determinan si la autenticación a implementar es parte de AH o ESP y el tipo de ésta autenticación.

- Los campos *left y right* son las direcciones IPv6 asignadas a cada gateway IPSec.

- Los campos *leftsubnet* y *rightsubnet* son las subredes que se encuentran detrás de cada gateway.

- Los campos *leftnexthop* y *rightnexthop* son las direcciones IPv6 del equipo que recibe la conexión.

Existe una sección “conn” especial y es la sección *conn %default*, esta sección va a tener los valores por defecto de todas las secciones “conn”. Luego en cada seccion “conn” se podrán sobreescribir los valores, pero en caso que se omitan se tomarán los escritos en esta sección.

Este es un ejemplo de sección *conn %default*

```
conn %default
keyingtries=0
authby=rsasig
```

El parámetro *keyingtries* determina cuantas veces *pluto* va a intentar establecer un túnel, el valor cero indica que siga intentando indefinidamente. El parámetro *authby* determina la forma en que se va a estar haciendo la autenticación. El valor *secret* es cuando se utiliza pre-shared-keys y se utiliza el valor *rsasig* cuando se va a autenticar con una firma digital RSA.

La Figura 67, retoma la topología de la red IPv6 que se está utilizando:

Figura 67 - Escenario de Prueba: Modo Transporte

En el escenario propuesto se debe tener en cuenta lo siguiente:

Modo Transporte AH/ESP	
Algoritmo de autenticación AH	hmac-md5
Algoritmo de autenticación ESP	hmac-md5
Algoritmo de cifrado ESP	3des-cbc

- En */usr/local/v6/etc/ipsec.conf* agregue una entrada **conn** transporte-ipv6 para establecer una sesión IPSec en modo transporte y configure la siguiente información:

config setup
....
conn transporte-ipv6
af=inet6
type=transport
auth=ah
authby=secret
left=3FFE:8070:1024:1::**1**
right=3FFE:8070:1024:1::**2**
esp=3des-md5-96
ah=hmac-md5-96

Una vez este archivo de configuración este salvado en los host**a** y host**b**, ejecute en cada uno de ellos lo siguiente:

- Ejecute *pluto* utilizando el comando: *#pluto,* o agregando las opciones *#pluto --nofork --stdrrlog --debug-all*, para efectos de depuración.

- Agregue la configuración de **conn** transporte-ipv6 a *pluto* ejecutando el comando: *#ipsec auto --add transporte-ipv6*

- Haga que *pluto* lea las llaves secretas: *#ipsec auto –ready*

- Haga que *pluto* establezca las AS's configuradas en *transporte-ipv6*: *#ipsec auto --up transporte-ipv6*

- Ejecute los comandos *#pfkey -L* o *#ipsec-conf show*, para desplegar la información de las asociaciones y de las políticas de seguridad establecidas.

- Guarde la configuración de las asociaciones y de las políticas de seguridad establecidas utilizando el comando *#ipsec-conf save <nombre-del-archivo>.*

P.5.5 Compare la salida de los archivos *<nombre-del-archivo>* y determine las diferencias que existen al configurar asociaciones y politicas de seguridad de forma manual y de forma automática.

P.5.6 Realice combinaciones: modo Transporte AH, modo Transporte ESP, modo Transporte AH/ESP de forma manual y automática y describa el proceso de encapsulación de los protocolos AH y ESP para su transporte sobre IPv6. ¿Qué puede concluir al respecto?

Fase 2: Habilitación de AH y/o ESP en modo Túnel

La Figura 68, describe la topología que se debe implementar:

Figura 68 - Escenario de Prueba

La configuración de las interfaces de red Ethernet en los nodos IPv6 es la siguiente:

	host**a**	GW de Seguridad **A**	GW de Seguridad **B**	host**b**
[eth0]	FEC0:0:0:1::X /64	3FFE:8070:1024:A:B:C::**1**/64	3FFE:8070:1024:A:B:C::**2**/64	FEC0:0:0:2::X /64
[eth1]	-	FEC0:0:0:1::**1**/64	FEC0:0:0:2::**2**/64	-
[tun6]	-	FE80::**1**/64	FE80::**2**/64	-

Tabla 20 - Direcciones IPv6

5.8 Procedimiento Fase 2

- Configuración Manual: Este tipo configuración no ofreció resultados exitosos, así que se decidió trabajar únicamente con la configuración automática, la cual ofreció los resultados esperados.

- Configuración Automática - IKE: En el escenario propuesto se debe tener en cuenta lo siguiente:

Modo Túnel AH/ESP	
Algoritmo de autenticación AH	hmac-md5
Algoritmo de autenticación ESP	hmac-md5
Algoritmo de cifrado ESP	3des-cbc

- Configure un dispositivo túnel entre las *gateways* de Seguridad **A** y **B**, y configure además enrutamiento. Para esto realice lo siguiente:

- En la *gateway* de Seguridad **A**

  - *#modprobe ipv6_tunnel*
  - *#ipv6tunnel add **tun6** --allow-local-packets encaplimit 0 remote 3FFE:8070:1024:A:B:C::**2** local 3FFE:8070:1024:A:B:C::**1***
  - *#ip link set **tun6** up*
  - *#ip addr add FE80::**1**/64 dev **tun6***
  - *#route -A inet6 add FEC0:0:0:2::/64 dev **tun6***

- En *gateway* de Seguridad **B**

  - *#modprobe ipv6_tunnel*
  - *#ipv6tunnel add **tun6** --allow-local-packets encaplimit 0 remote 3FFE:8070:1024:A:B:C::**1** local 3FFE:8070:1024:A:B:C::**2***
  - *#ip link set **tun6** up*
  - *#ip addr add FE80::**2**/64 dev **tun6***
  - *#route -A inet6 add FEC0:0:0:2::/64 dev **tun6***

- Utilizando ping6 envie tráfico ICMP del host**a** al host**b**.
- Utilice el analizador de protocolos de red Ethereal para identificar los tipos de mensajes que envía y recibe la *gateway* de Seguridad **B**. Para esto ejecute desde la consola de Linux el comando # *ethereal* sobre las interfaces de red eth0, eth1 y tun6.

P.5.7 Analice el campo dirección IPv6 origen y destino en cada uno de los casos: interfaces de red eth0, eth1 y tun6. Justifique su respuesta

P.5.8 Describa el proceso de encapsulación IPv6 en IPv6.

- Observe el campo *datos* del paquete UDP y tome nota de los valores en hexadecimal. Para esto guarde el archivo que genera *ethereal.*

- En */usr/local/v6/etc/ipsec.conf* agregue una entrada **conn** tunel-ipv6 para establecer una sesión IPSec en modo túnel.

```
config setup
....
conn tunel-ipv6
af=inet6
type=tunnel
auth=esp
authby=secret
left=3FFE:8070:1024:A:B:C::1
leftsubnet=FEC0:0:0:1::/64
leftnexthop=3FFE:8070:1024:A:B:C::2
right=3FFE:8070:1024:A:B:C::2
rightsubnet=FEC0:0:0:2::/64
rightnexthop=3FFE:8070:1024:A:B:C::1
esp=3des-md5-96
ah=hmac-md5-96
```

Una vez este archivo de configuración este salvado en los host**a** y host**b**, ejecute en cada uno de ellos lo siguiente:

- Ejecute *pluto* utilizando el comando: #*pluto,* o agregando las opciones #*pluto --nofork --stdrrlog --debug-all*, para efectos de depuración.

- Agregue la configuración de **conn** tunel-ipv6 a *pluto* ejecutando el comando: #*ipsec auto --add transporte-ipv6*

- Haga que *pluto* lea las llaves secretas: #*ipsec auto –ready*

- Haga que *pluto* establezca las AS's configuradas en *tunel-ipv6*: #*ipsec auto --up tunel-ipv6*

- Ejecute los comandos #*pfkey -L* o #*ipsec-conf show*, para desplegar la información de las asociaciones y de las políticas de seguridad establecidas.

- Envíe tráfico UDP desde el host**a** al host**b**.

- Utilice el analizador de protocolos de red Ethereal para identificar los tipos de mensajes que envía y recibe la *gateway* de Seguridad **B**. Para esto ejecute desde la consola de Linux el comando # *ethereal* sobre las interfaces de red eth0, eth1 y tun6.

- Analice el campo dirección IPv6 origen y destino en cada uno de los casos: interfaces de red eth0, eth1 y tun6. Justifique su respuesta

P.5.9 Describa los campos de las cabeceras AH y ESP.

- Observe el campo *datos* del paquete UDP y tome nota de los valores en hexadecimal. Para esto guarde el archivo que genera *ethereal.*

P.5.10 Describa el proceso de encapsulación de IPSec para su transporte sobre IPv6.

- Ahora que puede comparar el campo datos en los paquetes UDP recibidos, habilitando o deshabilitando una sesión IPSec entre las *gateways*, ¿Qué puede concluir al respecto?

5.9 **Análisis de Resultados**

5.10 **Conclusiones**

6 Laboratorio Internet IPv6: Calidad de Servicio - QoS

6.1 Motivación

Actualmente el desarrollo de redes de datos se esta enfocando hacia la provisión de Calidad de Servicio (QoS), la cual se requiere para permitir asegurar determinadas características de calidad en la transmisión de información. El objetivo es evitar que la congestión de determinados nodos de la red afecte a algunas aplicaciones que requieran un especial caudal o retardo, como pueden ser aplicaciones de videoconferencia. Para solucionar este problema existen dos tendencias bien distintas:

Sobredimensionar adecuadamente la red de transporte, lo que implica aumentar cuando resulte necesario los equipos de conmutación así como el ancho de banda disponible en los canales. Este método se basa en el abaratamiento de los sistemas de conmutación y transporte, si bien provoca una gestión ineficiente por definición de los recursos disponibles.

Gestionar de forma inteligente los recursos disponibles, compartiéndolo de manera desigual entre los diferentes flujos de tráfico. Sin embargo, las actuales redes de datos no distinguen entre las diferentes aplicaciones que transportan: no pueden diferenciar entre una videoconferencia con determinados requisitos de ancho de banda y la navegación web de características completamente diferentes. Esto requiere que de alguna manera las funciones de calidad de servicio sean capaces de reconocer las aplicaciones para reservarles unos determinados recursos en la red.

Considerando, que la segunda tendencia es la más óptima y teniendo en cuenta que una de las ventajas de IPv6 es permitir un manejo e implementación adecuada de Calidad de servicio haciendo uso de los campos clase de tráfico y etiqueta de flujo dentro de la cabecera IPv6 y de la diferenciación de servicios requeridos dentro de una red, en la siguiente práctica se realizará una implementación de QoS enfocada en el manejo del campo Clase de Tráfico de la cabecera IPv6, utilizando la herramienta Traffic Control sobre el S.O Linux Red Hat 9, por medio de la cual es posible obtener un control del tráfico dentro de una red enfocándose en la asignación de ancho de banda y prioridad de paquetes.

6.2 Objetivos

- Comprender los conceptos básicos sobre Calidad de Servicio en redes IPv6.

- Conocer la herramienta TC (Traffic Control) y las ventajas que tiene al momento de controlar tráfico haciendo énfasis en la diferenciación de servicios.

- Implementar QoS utilizando la herramienta TC dentro de una topología de red IPv6.

- Manejar colas, clases y filtros con el fin de controlar los diferentes tipos de paquetes que circulan por la red piloto UniCauca IPv6.

6.3 Marco teórico

6.3.1 Calidad del servicio en IPv6

En términos generales, puede definirse la Calidad del Servicio (QoS) como la capacidad que tiene un sistema de asegurar, con un grado de fiabilidad preestablecido, que se cumplan los requisitos de tráfico, en términos de perfil y ancho de banda, para un flujo de información dado. Más específicamente, para el caso de proveedores de red, se establece en el RFC 2475 (An Architecture for Differentiated Services) donde un servicio define algunas características significativas de la trasmisión de un paquete en una dirección, a través de un conjunto de una o más rutas dentro de la red. Estas características pueden especificarse en términos de caudal (throughput), demora (delay), variación de demora (jitter) y/o pérdidas, o también en términos de alguna prioridad relativa de acceso a los recursos de la red.

En este momento existen principalmente dos tipos de tecnologías que proporcionan calidad de servicio. La primera se basa en la **reserva**, y asigna recursos basándose en flujos de tráfico. Alternativamente, un segundo tipo de calidad de servicio se caracteriza por la **priorización** de determinado tipo de tráfico. Los flujos de datos individuales se van agrupando en grandes agregados de tráfico de acuerdo a la *Clase de Servicio* a la que pertenezcan, y dependiendo de esa Clase de Servicio recibirán un distinto trato en los diferentes elementos de la red.

En comunicaciones IP se traduce en dos modelos de trabajo:

6.3.1.1 Modelo Intserv - Arquitectura de Servicios Integrados (ISA)

Basado en el protocolo RSVP *(Resource ReSerVation Protocol, RFC 1633)*, implica una reserva de recursos en la red para cada flujo de información de usuario, así como el mantenimiento en la red (en los enrutadores*)* de un estado para cada flujo, esto es, mantenimiento de la reserva - *tablas de estados de reserva* -. Esto conduce a un considerable tráfico de señalización y ocupación de recursos en cada enrutador para cada flujo, con la consiguiente complejidad en el hardware, al margen del aporte que esta señalización hace a la congestión de la red. No es una solución escalable, no es una solución adecuada para grandes entornos como Internet.

Por medio de este método, la calidad de servicio puede ser garantizada, pero los Servicios Integrados son complejos de implementar y pueden generar mucho tráfico de señalización, lo cual lleva a problemas de escalabilidad.

RSVP es un protocolo señalización de QoS, y posibilita:

- Dar a las aplicaciones un modo uniforme para solicitar determinado nivel de QoS
- Encontrar una forma de garantizar cierto nivel de QoS
- Autenticación

RSVP se desarrolla entre los usuarios y la red, y entre los diferentes nodos (*enrutadores*) de la red que soportan este protocolo. Consiste en hacer *reservas* de recursos en dichos nodos para cada flujo de información de usuario, con la consecuente ocupación de los mismos. Esto requiere, lógicamente, intercambio de mensajes RSVP entre dichos entes funcionales, así como *mantener* estados de reserva en cada nodo RSVP. De manera que tanto la solicitud de las reservas, como el mantenimiento de éstas durante la comunicación, y la posterior cancelación, implican el intercambio de mensajes de señalización, lo que representa un tráfico considerable cuando de entornos en Internet se trata.

RSVP ofrece dos tipos de servicios:

- Servicio de carga controlada: se entiende en general que la pérdida de paquetes debe ser muy baja o nula.

- Servicio garantizado: se basa en solicitar determinado ancho de banda y cierta demora de tránsito máxima.

De los dos tipos de servicios que RSVP soporta, el más adecuado para aplicaciones con requerimientos de tiempo real es el servicio garantizado, aunque es más complejo de implementar que el servicio de carga controlada.

6.3.1.2 **Modelo Diffserv**

Los servicios diferenciados (Diffserv[82]) proporcionan mecanismos de Calidad de Servicio para reducir la carga en dispositivos de la red a través de un mapeo entre flujos de tráfico y niveles de servicio. Los paquetes que pertenecen a una determinada clase se marcan con un código específico (DSCP - Diffserv CodePoint). La diferenciación de servicios se logra mediante la definición de comportamientos específicos para cada clase de tráfico entre dispositivos de interconexión, hecho conocido como PHB (Per Hop Behavior).

[82] RFC 2475 An Architecture for Differentiated Service

Este modelo se basa en la división del tráfico en diferentes clases y en la asignación de prioridades a estos agregados, realizando la separación de los conceptos básicos de operación de los enrutadores de reenvío y control. En el reenvío se realiza un tratamiento diferenciado de los paquetes, de acuerdo con la Clase de Tráfico. Utiliza diferente información de la cabecera de los paquetes (por ejemplo, DSCP – Diffserv Code Point) para distinguirlos, clasificarlos y conocer el tratamiento que debe recibir el tráfico en los nodos de la red Diffserv. Es más simple que Intserv y se escala mejor.

En la Cabecera del paquete IPv6, existe un campo llamado Clase de Tráfico (8 bits); en la Cabecera del paquete IPv4, este campo esta en el área TOS (Tipo de servicio). Clase de Tráfico y TOS son renombrados como campo DS dentro del contexto Diffserv, el valor del código esta referido al código DSCP como se muestra en la Figura 69. El campo DS tiene seis bits para el código DSCP y dos bits que normalmente no se utilizan.

En Diffserv, los nodos de los extremos de una red clasifican el tráfico y fijan el DSCP (valor Diffserv). Los nodos externos son también responsables de mantener o configurar el tráfico. En el núcleo de la red, el tráfico es reenviado basándose sobre un PHB asociado con un particular DSCP.

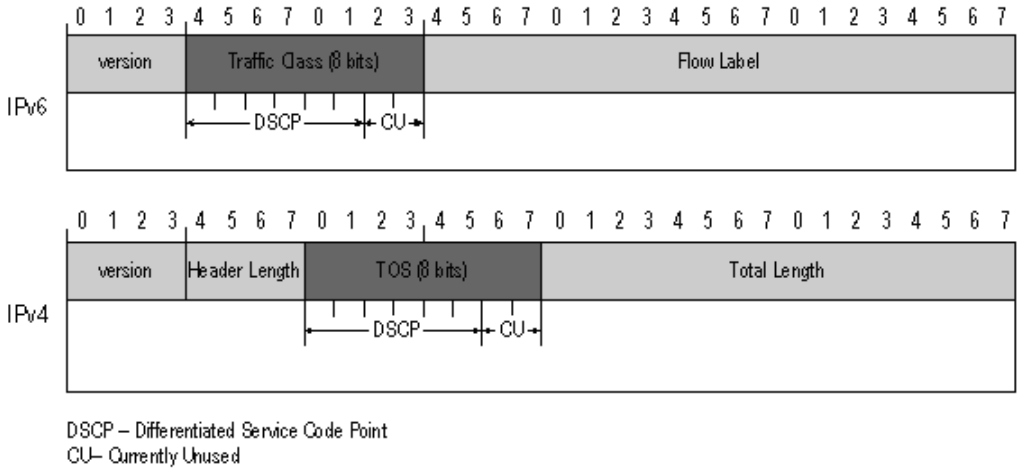


Figura 69 - DSCP en Cabeceras IPv6 e IPv4

Cada código mapea un PHB determinado, actualmente existen 3 PHB especificados para ser usados dentro de una red de servicios diferenciados:

- Comportamiento por omisión (Default Behavior)
- Tránsito acelerado (Expedited forwarding)
- Tránsito asegurado (Assured forwarding)

Comportamiento por omisión (mejor esfuerzo)

Es el comportamiento que todas las redes que implementen DiffServ deben de incorporar. Este comportamiento equivale a un servicio de mejor esfuerzo. Todos los paquetes que no tengan especificado un comportamiento, utilizan el servicio de mejor esfuerzo para moverse a través de la red. El código que representa el comportamiento por omisión es el 0x000000.

Tránsito acelerado

Este PHB tiene asociado una tasa de transmisión la cual la define el ISP. La función de este PHB es proveer las herramientas necesarias para prestar un servicio punto a punto con bajas pérdidas, bajo retardo, bajo jitter y un ancho de banda asegurado dentro de un dominio DiffServ. El valor del subcampo DSCP relacionado con este servicio es 101110.

La Figura 70 nos muestra el principio de operación del EF[83] PHB.

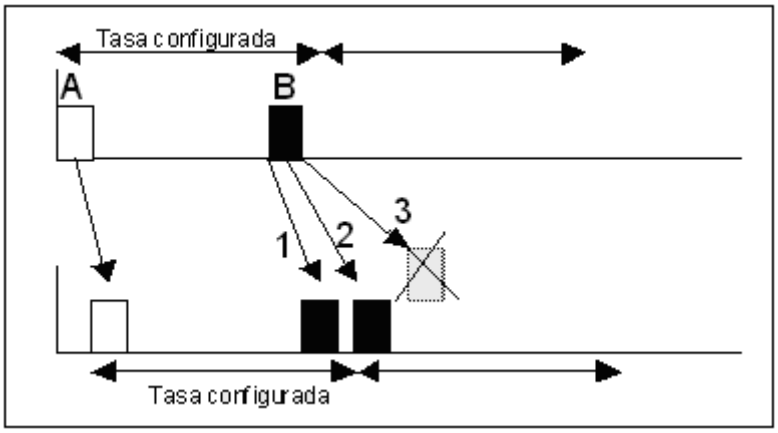


Figura 70 - Modelado y descarte de paquetes

Cuando el paquete llega antes de su tiempo programado de llegada, existen tres opciones en los nodos de ingreso e internos para su tratamiento:

- Reenviar el paquete inmediatamente
- Reenviar el paquete en el tiempo configurado
- Descartar el paquete

Las opciones que toman los nodos de acceso e internos son diferentes: los nodos de acceso por lo general tomaran las opciones 2 y 3 para evitar que la fuente se apropie de un mayor ancho de banda del que se tiene configurado.

[83] RFC 2598 - An Expedited Forwarding PHB

Para los nodos internos, es altamente recomendada la opción 1, ya que la aplicación de la opción 2 podría provocar retardos acumulados.

El EF PHB requiere un alto control sobre la tasa de transmisión de paquetes en los nodos de acceso a la red y de un rápido reenvío de paquetes en los nodos internos de la red.

De la especificación y de lo anteriormente visto se puede definir la región de acción en el modelo de servicio. Ya que es necesario un estricto control de la tasa de transmisión, el servicio ofrecido por este PHB es muy similar al de una línea dedicada.

La finalidad de este PHB es la de proveer enlaces de alta calidad, con respecto a retardo y pérdidas. EF puede ser utilizado para proveer enlaces que simulen enlaces dedicados, con bajos retardos y bajas variaciones en el ancho de banda.

Transito asegurado

Este PHP define 4 clases, a las cuales se les tiene que asignar espacio en el buffer y ancho de banda de manera independiente en cada nodo. A cada una de estas clases se le especifica 3 niveles de descarte. Es importante señalar que no es necesario implementar los 3 niveles de descarte. Si el administrador de la red, no espera que existan muchas condiciones de congestión, el número de niveles de descarte se puede compactar a 2.

	Clase 1	**Clase 2**	**Clase 3**	**Clase 4**
Baja probabilidad de descarte	001010	010010	011010	100010
Media probabilidad de descarte	001100	010100	011100	100100
Alta probabilidad de descarte	001110	010110	011110	100110

Tabla 21 - Códigos DS recomendados para AS PHB

6.3.1.3 **Componentes de DiffServ**

Clasificadores

Existen 2 tipos de clasificadores en DiffServ:

- Clasificadores multicampo: Generalmente existen únicamente para los extremos de la red, y son utilizados para clasificar paquetes basados sobre múltiples valores en la cabecera (como dirección de origen, protocolo, número de puerto ..). El propósito de este tipo de clasificación es fijar un valor apropiado para el DSCP basado sobre un criterio configurado administrativamente.

- Clasificador de comportamiento agregado (BA): Únicamente clasifica paquetes basados en DSCP, el clasificador BA es utilizado para el control PHB.

Acondicionador de Tráfico

Un acondicionador de tráfico consiste de un medidor de tráfico, un modelador, un eliminador y un marcador.

- Medidor de tráfico (*Traffic Meter*): Mide las propiedades temporales de los paquetes.

- Conformador (*Shapers*): establece cierta demora para uno o más paquetes de un flujo.

- Eliminador: descarta algunos o todos los paquetes de un flujo de tráfico.

Marcador de paquetes **(*Packet Markers*)**

Establece un *codepoint* en el campo. El acondicionador es capaz de comparar perfiles de un flujo clasificado contra el perfil esperado utilizando un metro. El metro puede ser utilizado para alterar la formación o el rechazo de paquetes. La formación y el rechazo son usualmente implementados dentro del sheduler QoS. El marcador de tráfico es utilizado para fijar el DSCP del paquete basado en la clasificación del paquete y la entrada del medidor de tráfico. El valor del DSCP afectará la conformación del paquete en los nodos siguientes.

Programador de Tráfico

En el caso de DiffServ, el programador decide cual paquete se envía y cuando enviar el próximo, basado en resultados de la clasificación y observación del perfil de tráfico para cada clase.

6.3.1.4 **Los tipos de enrutadores en redes *Diffserv***

First Hop Router: es el enrutador más próximo al *host* emisor de paquetes. Los flujos de paquetes son clasificados y marcados acorde a la etiqueta SLA (*Service Level Agreement*). Es responsable de que el tráfico esté acorde con el ancho de banda del perfil.

Ingrees Router: se sitúan en los puntos de entrada al *backbone Diff-Serv* (dominio DS), efectuando la clasificación de los paquetes en base al campo DS o en base a múltiples campos de la cabecera de éstos.

Egress Router: se ubican en los puntos de salida de redes Diff-Serv (dominio DS), controlando el tráfico. Efectúan la clasificación de paquetes en base solo al campo DS de las cabeceras.

Interior router: tienen la misión de «sumar» flujos, realizar la clasificación DS y reenvio de paquetes. Se sitúan dentro del *backbone* DS (dominio DS).

Los nodos intermedios (enrutadores) tendrán que analizar estos paquetes y tratarlos según sus necesidades. Esta es la razón principal por la que Diffserv ofrece mejores características de escalabilidad que Intserv. Dentro del grupo de trabajo de Diffserv de la IETF, se define en el campo DS (Differentiated Services) donde se especificarán las prioridades de los paquetes.

En el subcampo DSCP (Differentiated Service CodePoint) se especifica la prioridad de cada paquete. Estos campos son validos tanto para IPv4 como IPv6.

6.3.2 **Concepto de Cola y disciplina de Colas**

Concretamente, dentro del campo de la Gestión de Ancho de Banda, Linux ofrece amplias posibilidades.

Internet utiliza la pila de protocolos TCP/IP. Sin embargo, TCP/IP no tiene forma de conocer la capacidad (en ancho de banda) de la red que une dos equipos que se comunican. De esta forma lo que ocurre es que los paquetes son enviados cada vez más rápido hasta que empieza a perderse parte de los mismos debido a que se supera la capacidad de la red, momento que TCP/IP aprovecha para ajustar la velocidad de envío.

Esta forma de trabajar es lo que se puede controlar y modificar mediante el proceso de encolar paquetes. Las colas y disciplinas de colas son, junto con otras, las herramientas que permiten definir las políticas de los procesos de encolar paquetes.

Disciplina de colas

Es el algoritmo que gestiona el proceso de encolar paquetes en un dispositivo (interfaz de red). Esta gestión puede ser tanto en la cola de entrada (ingress), como en la cola de salida (egress).

6.3.2.1 **Disciplinas de colas sin clases**

Es aquella disciplina de colas que no admite una subdivisión interna que pueda ser configurada por el usuario.
Estos son los procesos de encolar paquetes más sencillos, debido a que sólo tienen la capacidad de reordenar, retrasar o descartar los paquetes que van llegando para ser enviados:

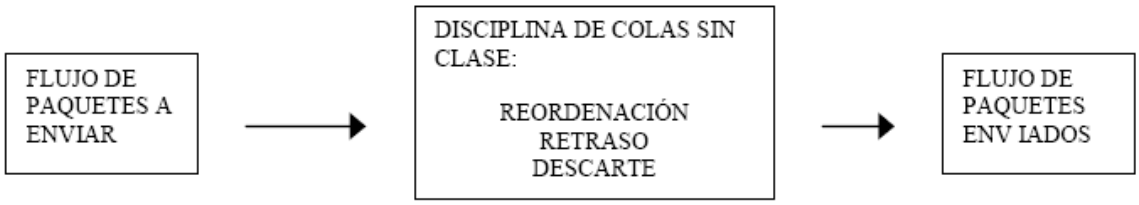


Figura 71 - Esquema disciplina de colas sin clases

Los tres procesos de encolar paquetes que existen de este tipo son:

Pfifo Fast

Esta disciplina de colas está formada por tres bandas (bandas 0, 1 y 2). Dentro de cada banda los paquetes son enviados siguiendo una política FIFO (First In, First Out). Sin embargo, ningún paquete de la banda 1 es enviado mientras existan paquetes por enviar en la banda 0, y lo mismo ocurre para las bandas 2 y 1. Es decir, existe una prioridad definida entre dichas bandas, siendo la banda 0 la más prioritaria y la banda 2 la de menor prioridad.

En la definición de disciplina de colas sin clases se especificó que éstas no tenían ninguna subdivisión interna en su estructura. Sin embargo, esta disciplina pfifo_fast posee tres bandas. Una disciplina de colas sin clases no puede tener ninguna subdivisión interna de su estructura, susceptible de ser configurada por el usuario. Así pues, aunque la pfifo_fast tiene subdivisión interna, esta no puede ser modificada por el usuario.

Token Bucket Filter

Este tipo de disciplina de colas es la que se debe escoger en el caso de que la única necesidad sea limitar el ancho de banda de un determinado interfaz.

El modelo de funcionamiento consiste en suponer que se tiene un buffer (bucket) al cual llegan los denominados ‘tokens’ a un ritmo constante. Estos tokens además serán los que utilizarán los paquetes IP para salir del interfaz de red. Es como si cada paquete IP tuviese que esperar a una carretilla (token) que será la encargada de sacarlo del interfaz.

Este modelo es un poco simple aunque explica perfectamente la dinámica de este algoritmo de disciplina de colas. En realidad, los tokens tienen correspondencia con bytes y no con paquetes IP, pero el resto del modelo es bastante aproximado.

Stochastic Fairness Queueing

Este tipo de disciplina de colas intenta distribuir el ancho de banda de un determinado interfaz de red de la forma más justa posible.

Para ello esta disciplina implementa una política de Round Robin entre todos y cada uno de los flujos de comunicación establecidos en el interfaz, dando a cada uno la oportunidad de enviar sus paquetes por turnos. Un flujo de comunicación será cualquier sesión TCP o flujo UDP, y de esta forma lo que se logra es que ninguna comunicación impida al resto poder enviar parte de su información. Lógicamente esta disciplina de colas sólo tendrá sentido en aquellos interfaces que normalmente estén saturados y en los que no queramos que una determinada comunicación eclipse al resto.

6.3.2.2 **Disciplinas de colas con clases**

Una disciplina de colas con clases puede contener muchas clases, cada una de las cuales es interna a la disciplina de colas. Además, cada clase contiene una nueva disciplina de colas, que puede ser con clases o sin ellas.

Este tipo de disciplinas de colas se caracterizan por tener una subdivisión interna de su estructura, susceptible de ser configurada por el usuario, lo cual las hace muy útiles cuando se tienen diferentes tipos de tráfico que necesitan diferentes tratamientos.

Cuando los paquetes IP llegan a una disciplina de este tipo, necesitan ser enviados a una de las clases que la componen, es decir, necesitan ser clasificados. Para realizar esta clasificación se consultan los filtros asociados a la disciplina de colas, los cuales devuelven un resultado que permite a la disciplina de colas determinar a qué clase debe ser enviado el paquete. Además, se conoce que cada clase tiene asociada una nueva disciplina de colas (con o sin clases), con lo que nuevas consultas a filtros pueden ser realizadas hasta conseguir clasificar el paquete completamente.

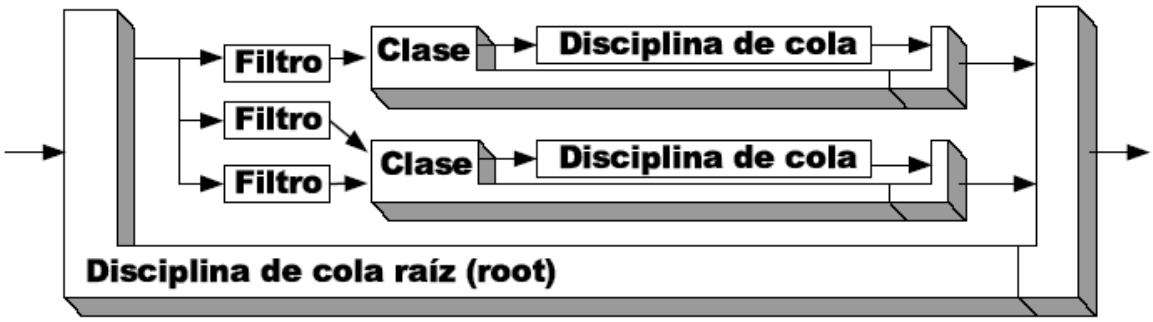


Figura 72 - Esquema Disciplina de Colas con Clases

En Linux, cada interfaz de red tiene una disciplina de colas de salida (egress) llamada 'root', que es la primera de su estructura interna. Por defecto, si no se especifica otra cosa, esta disciplina es del tipo pfifo_fast. Además, a cada disciplina de colas le es asignado un 'manejador' que se utilizará en los comandos de configuración de dicha disciplina. Estos manejadores constan de dos partes, un 'número mayor' y un 'número menor' separados por ':', así el manejador de la disciplina de colas 'root' es '1:0'. Normalmente el número menor del manejador de una disciplina de colas es siempre cero, y el número mayor de las clases adjuntas a una disciplina de colas debe coincidir con el número mayor de la misma.

Disciplina de colas PRIO

Por defecto esta disciplina de colas define tres clases, y cada una de ellas tiene asociada una nueva disciplina de colas con política FIFO. Entre las tres clases existe una prioridad de forma que mientras haya paquetes en la clase 1 no se envían paquetes de la clase 2, y lo mismo entre las clases 2 y 3.

Parámetros de las disciplinas de colas PRIO

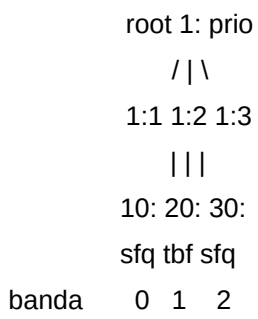
En este tipo de disciplinas se pueden configurar dos parámetros:

- *Bands:* Este parámetro permite especificar el número de clases (por defecto 3) que se quiere que tenga la disciplina.

- *Priomap:* En una disciplina de este tipo es posible decidir no adjuntar ningún filtro que realice la clasificación del tráfico entre las distintas clases que se tengan. En ese caso la clasificación se hace siguiendo el mismo criterio (campo TOS) que en el caso de la disciplina pfifo_fast.

Un ejemplo de utilización sería el siguiente:

Se supone un interfaz en el que se quiere dar prioridad al tráfico que necesita interactividad, frente al resto de tráfico. Para ello se crearía la siguiente estructura en la disciplina de colas:



El tráfico normal es enviado a la clase 30:, el tráfico que necesita interactividad es enviado a las clases 20: o 10:.

Disciplina de cola FIFO (First In First Out)

Es la cola utilizada por defecto cuando se activa una interfaz. Se implementa cuando simplemente se necesita una cola.

Paquete FIFO

tc qdisc add dev STRING [handle QHANDLE] [root | parent CLASSID] pfifo limit LIMIT
Parámetros: limit LIMIT: Número máximo de paquetes en cola

Disciplina de colas Class-Based Queueing (CBQ)

CBQ es la más compleja y más empírica de todas las disciplinas de colas. Esto obedece básicamente al hecho de que, además de ser una disciplina de colas con clase, CBQ ofrece la capacidad de regular el ancho de banda (shaping), siendo en este terreno donde surgen sus mayores dificultades.

Se supone que se intenta reducir el ancho de banda de una conexión de red de 10 Mbit/seg a 1Mbit/seg. Para lograrlo se tendrá que conseguir que el enlace esté inactivo el 90% del tiempo. Sin embargo, medir con exactitud ese porcentaje entraña mucha dificultad.

CBQ utiliza un algoritmo basado en la estimación del intervalo de tiempo transcurrido entre dos peticiones consecutivas del hardware para el envío de datos. Así, se ha comprobado que no

siempre se consiguen buenas aproximaciones de este modo, e incluso a veces los resultados son totalmente equívocos.
No obstante, para la mayoría de las situaciones, este algoritmo trabaja de forma adecuada cumpliendo perfectamente con nuestras necesidades.

Parámetros CBQ para ajustar la regulación del ancho de banda.

CBQ trabaja intentando que el enlace esté inactivo durante un porcentaje de tiempo que asegure que se regula hasta conseguir el ancho de banda deseado.

Algunos de los parámetros más importantes son:

- *Avpkt:* Tamaño medio del paquete medido en bytes. Se necesita para calcular maxidle, que se deriva de maxburst, que va especificado en paquetes.

- *Bandwidth:* Ancho de banda del dispositivo físico. Se necesita para calcular el tiempo muerto entre petición y petición.

- *Cell*: El tiempo que tarda un paquete en ser transmitido sobre un dispositivo está escalonado, y se basa en el tamaño del paquete. Un paquete de tamaño 800 y uno de 806 pueden tardar lo mismo en ser enviados, por ejemplo (esto establece la granularidad). A menudo está a "8". Debe ser una potencia entera de dos.

- *Maxburst*: Número de bytes que sera´n enviados en el burst más grande posible.Este número de paquetes se usa para calcular maxidel de manera que cuando avgidle esté a maxidel, se puede enviar una ráfaga de esta cantidad de paquetes medios antes de que avgidle caiga a 0. Póngalo alto si quiere que sea tolerante a ráfagas. No puede establecer maxidel directamente, sino sólo mediante este parámetro.

- *MTU:* Tamaño mínimo de un paquete. Es necesario porque incluso un paquete de cero bytes de datos da lugar a una trama Ethernet de un tamaño mínimo distinto de cero.

- *Rate.* Ancho de banda regulado con el que se quiere que funcione la disciplina de colas configurada

Parámetros CBQ para definir el comportamiento como disciplina de colas con clases.

Al igual que la disciplina PRIO, CBQ permite definir prioridades dentro de las clases que componen su estructura interna. De esta forma, cada vez que el nivel hardware solicita un

paquete, CBQ lanza un proceso Round Robin por prioridades. Para la configuración de este proceso CBQ pone a disposición del usuario los siguientes parámetros:

- *Prio:* Establece las distintas prioridades entre las clases que componen la estructura interna de la disciplina de colas.

- *Allot:* Cuando se le pide a la CBQ externa un paquete para enviar por la interfaz, buscará por turnos en todas sus qdisc internas (en las clases), en el orden del parámetro de «prioridad». Cada vez que le toca el turno a una clase, sólo puede enviar una cantidad limitada de datos. «Allot» es la unidad básica de esta cantidad. Vea el parámetro «weight» para más información.

- *weight*: Weight ayuda en el proceso de Weighted Round Robin. Cada clase tiene una oportunidad por turnos para enviar. Si tiene una clase con un ancho de banda significativamente menor que las otras, tiene sentido permitirle enviar más datos en su ronda que a las otras.

Una CBQ suma todos los pesos bajo una clase, y los normaliza, de manera que puede usar números arbitrarios: sólo son importantes las equivalencias. La gente viene usando «tasa/10» como regla general y parece que funciona bien. El peso renormalizado se multiplica por el parámetro «allot» para determinar cuántos datos se envían en una ronda.

Parámetros CBQ que determinan la compartición y préstamo de enlaces

Aparte de limitar determinados tipos de tráfico, también es posible especificar qué clases pueden tomar prestada capacidad de otras clases o, al revés, prestar ancho de banda.

- *Isolated/sharing:* Una clase que está configurada como «isolated» (aislada) no prestará ancho de banda a sus hermanas. Uselo si tiene varios agentes competidores o mutuamente hostiles sobre el enlace que no quieren dejarse espacio entre sí. El programa de control tc también conoce el «sharing», que es lo contrario que «isolated».

- *bounded/borrow* :Una clase también puede estar «bounded» (limitada), lo que significa que no tratará de tomar ancho de banda prestado de sus clases hermanas. tc también conoce «borrow», que es lo contrario de «bounded».

- En una situación típica sería posible encontrar a dos agentes sobre un mismo enlace que al mismo tiempo son «isolated» y «bounded», lo que significa que están realmente limitadas a sus tasas aisladas, y que no permitirán préstamos entre sí. Junto a tales clases, puede haber otras que tengan permiso de ceder ancho de banda.

Cola RED

La disciplina de encolado RED calcula el tamaño promedio de las colas. Este tamaño es comparado con dos umbrales, un mínimo y un máximo. Cuando el promedio de tamaño de las colas es menor que el mínimo los paquetes no son marcados. Cuando el promedio establecido es mayor que el máximo, los paquetes son marcados y luego son desechados. Cuando el promedio está entre el mínimo y el máximo umbral, cada paquete que llega es marcado con una probabilidad Pa en donde pa es una función del promedio de tamaño de cola avg.

RED descarta paquetes de los flujos estadísticamente antes de que lleguen a un límite absoluto (hard). Esto hace que un enlace congestionado en un backbone reduzca la marcha de una manera más elegante, y evita la sincronización de retransmisiones.

También ayuda a TCP a encontrar su velocidad "correcta" más rápido permitiendo que algunos paquetes caigan pronto manteniendo bajo el tamaño de las colas y la latencia bajo control. La probabilidad de que se descarte un paquete de una conexión particular es proporcional a su uso de ancho de banda en lugar de al número de paquetes que transmite.

Comandos TC para una cola RED

tc qdisc add dev STRING [handle QHANDLE] [root | parent CLASSID] red limit BYTES bandwidth BPS min BYTES max BYTES avpkt BYTES burst PKTS probability FLOAT

Parámetros:

- limit BYTES: Tamaño fisico de la cola
- bandwidth BPS: El máximo ancho de banda para la interfaz a la cual se adjunta la cola
- min BYTES: Mínimo umbral
- max BYTES: Máximo umbral
- avpkt BYTES: Número promedio de bytes en un paquete en donde va esta clase.
- burst PKTS: Utilizado para calcular el tiempo de restricción.
- probability FLOAT: Probabilidad de paquetes desechados aleatoriamente

Disciplina de colas Hierarchical Token Bucket (HTB)

Este tipo de qdisc esta disponible oficialmente en el kernel desde la versión 2.4.20. Debido a la facilidad de uso y configuración, este tipo de qdisc esta siendo muy utilizado, sin embargo, HTB aún no es muy difundida. HTB es muy similar a CBQ, pero no efectúa cálculos de tiempo ocioso para los ajustes, sino que es un Token Bucket Filter con clases.

Parámetros:

- rate: Es la tasa de tráfico que se desea que salga de la cola.
- burst: Especifica el tamaño máximo del bucket en bytes.
- default: Específica a que cola irán por defecto los paquetes que no son clasificados.
- ceil: Especifica el ancho de banda máximo que una clase puede utilizar.
- prio: Establece la prioridad de las clases, primero se busca en las clases de menor prioridad y mientras éstas tengan tráfico no se procesas las otras clases.

Qdisc: dsmark

Dsmark es una disciplina de colas que ofrece las capacidades necesarias para los Servicios Diferenciados (también conocidos como DiffServ o simplemente DS).
Como especifica la bibliografía de DiffServ, se diferencia entre nodos externos (o limítrofes) e internos. Son dos puntos importantes en el camino del tráfico. Ambos tipos realizan una clasificación cuando llega el paquete. Su resultado puede usarse en diferentes lugares a lo largo del proceso de DS antes de que se envié el paquete a la red.
Es por esto que el código de diffserv proporciona una estructura llamada sk_buff, que incluye un campo nuevo llamado skb->tc_index donde se almacenará el resultado de la clasificación inicial que puede usarse en varios momentos del tratamiento DS.
El valor de skb->tc_index lo establecerá inicialmente la qdisc DSMARK, sacándola del campo DS de la cabecera IP de cada paquete recibido. Aparte, el clasificador cls_tcindex leerá todo o parte del valor de skb->tcindex y lo usará para escoger las clases.

Este tipo de cola se utiliza si se desea controlar tráfico utilizando el campo DS en la cabecera IP

tc qdisc add dev STRING [handle QHANDLE] [root | parent CLASSID] dsmark indices INDICES [default_index DEFAULT_INDEX] [set_tc_index]

Parámetros:
indices INDICES: Tamaño de la tabla (mascara y valor). El Maximo valor es 2^n, donde n>=0.
default_index DEFAULT_INDEX: Indice de entrada de la tabla por defecto si no se encuentra un clasificador marcado.
set_tc_index: Ordena a la disciplina dsmark a recuperar el campo DS y almacenarlo en skb->tc_index

Funcionamiento de la cola dsmark

Si se ha declarado la opción set_tc_index en la orden de la qdisc, se obtiene el campo DS y se almacena en la variable skb->tc_index.
Se invoca al clasificador. Será ejecutado y devolverá un ID de clase que será almacenado en la variable skb->tc_index. Si no se encuentra ningún filtro que concuerde, se optará por la opción default_index como classId a almacenar. Si no se ha declarado ni set_tc_index ni default_index el resultado puede ser impredecible.
Después de ser enviado a las qdisc internas donde se puede reutilizar el resultado del filtro, se almacena el classid devuelto por la qdisc interna en skb->tc_index. Se utilizará este valor en adelante para indicar una tabla máscara-valor. El resultado final a asignar al paquete será el que resulte de la siguiente operación:

Nuevo_Campo_Ds = (Viejo_Campo_Ds & máscara) | valor

Por tanto, el nuevo valor saldrá de aplicar AND a los valores ds_field y máscara, para después aplicar al resultado un OR con el parámetro valor.

UTILIZACIÓN DE FILTROS PARA LA CLASIFICACIÓN DE PAQUETES.

Clasificador y filtro

Las disciplinas de colas con clases necesitan determinar a qué clase envían cada paquete que llega. Esto lo hacen utilizando un clasificador. A su vez la clasificación la llevan a cabo utilizando filtros, los cuales determinan una serie de condiciones que deben cumplir los paquetes.

Como se ha dicho anteriormente, en las disciplinas de colas con clases es necesario llevar a cabo una clasificación del tráfico. Es decir, es necesario determinar a qué clase debe ir cada paquete IP.
Para ello se utiliza el concepto de ‘filtro’ que será asociado a cada disciplina de colas en la que sea necesario llevar a cabo una clasificación.

Filtro u32.

Este tipo de filtro proporciona una versatilidad enorme al permitir muchos criterios a la hora de llevar a cabo el filtrado, lo cual hace que sean los más ampliamente utilizados.
Por definición, este tipo de filtro permite filtrar en función de cualquier conjunto de bits, tanto de la cabecera del paquete IP, como de la cabecera del segmento de datos.

Sin embargo, este tipo de utilización es bastante farragosa y complicada, por lo que normalmente se suelen utilizar formas más directas para estos filtros. Así, algunas de estos criterios directos son:

- Dirección IP de origen/destino del paquete.
- Protocolo utilizado: tcp, udp, icmp, gre.
- Puertos de origen y destino utilizados.
- Valor del campo TOS de la cabecera IP.

Sintaxis filtro U32

tc filter add dev STRING [pref PRIO] [protocol PROTO] [estimator INTERVAL TIME_CONSTANT] [root | classid CLASSID] [handle FILTERID] u32 [match SELECTOR ...] [link HTID] [classid CLASSID] [police POLICE_SPEC] [sample SAMPLE] or u32 divisor DIVISOR

Parámetros:

- handle FILTERID: Identificador del filtro X:Y:Z
- classid CLASSID
- police POLICE_SPEC
- offset OFFSET_SPEC
- ht HTID: tabla hash
- hashkey HASHKEY_SPEC: esta compuesto por:
- mask MASK
- VALUE
- divisior DIVISIOR : número de entradas de la tabla hash
- match SELECTOR: Regla de marcado.
- Otros parámetros u32
- Burst: Tamaño del bucket en bytes.
- Drop: descarta el tráfico que excede de una cierta tasa

Filtro Tcindex

tc filter add dev STRING [pref PRIO] [protocol PROTO] [estimator INTERVAL TIME_CONSTANT] [root | classid CLASSID] [handle FILTERID] tcindex [hash SIZE] [mask MASK] [shift SHIFT] [pass_on | fall_through] [classid CLASSID] [police POLICE_SPEC]

Parametros:

- hash SIZE
- mask MASK
- shift SHIFT
- pass_on

- fall_through
- classid CLASSID
- police POLICE

Con el fin de explicar el funcionamiento de este filtro, a continuación se da un ejemplo de configuración utilizando una cola dsmark con un filtro tcindex:

tc qdisc add dev eth0 handle 1:0 root dsmark indices 64 set_tc_index
tc filter add dev eth0 parent 1:0 protocol ip prio 1 tcindex mask 0xfc shift 2 tc qdisc add dev eth0 parent 1:0 handle 2:0 cbq bandwidth 10Mbit cell 8 avpkt 1000 mpu 64 #

clase de tráfico EF
tc class add dev eth0 parent 2:0 classid 2:1 cbq bandwidth 10Mbit rate 1500Kbit avpkt 1000 prio 1 bounded isolated allot 1514 weight 1 maxburst 10 &num
qdisc fifo de paquetes para tráfico EF tc qdisc add dev eth0 parent 2:1 pfifo limit 5
tc filter add dev eth0 parent 2:0 protocol ip prio 1 handle 0x2e tcindex classid 2:1 pass_on.

(Este código no está completo. Es sólo un extracto del ejemplo EFCBQ incluido en la distribución de iproute2).

Se supone que se recibe un paquete marcado como EF (expedited forwarding). Si lee el RFC2598, verá que el valor DSCP recomendado para el tráfico EF es 101110. Esto significa que el campo DS será 101110 o 0xb8 en código hexadecimal.

Entonces llega el paquete con el valor del campo DS puesto a 0xb8. Como se explicó anteriormente, la qdisc dsmark toma el campo DS y lo almacena en la variable skb->tc_index. El siguiente paso del ejemplo corresponde al filtro asociado con esta qdisc (segunda línea del ejemplo). Esto realizará las siguientes operaciones:

Valor1 = skb->tc_index & MASK
Clave = Valor1 >> SHIFT

En el ejemplo, MASC=0xFC i SHIFT=2.
Valor1 = 10111000 & 11111100 = 10111000
Clave = 10111000 >> 2 = 00101110 -> 0x2E en hexadecimal
Los últimos parámetros que hay que comentar son hash y pass_on. El primero se refiere al tamaño de la tabla hash. Pass_on se usa para indicar que si no se encuentra un classid igual al resultado de este filtro, se debe intentar con el siguiente filtro. La acción por defecto es fall_through.

Filtro route

Este tipo de filtros toman su decisión en función del resultado obtenido al pasar el paquete IP por la tabla de rutas.

6.3.2.3 **ARQUITECTURA DE DIFFSERV LINUX**

El código Diffserv ha sido integrado dentro del kernel de linux-2.4x. Las características Diffserv pueden ser habilitadas fácilmente editando los archivos de configuración de TC.

La función del control de tráfico en linux se observa en la Figura 73. Paquetes entrantes son examinados y son directamente reenviados por la red (por ejemplo sobre una interfaz diferente, si la máquina está actuando como un enrutador), o son transmitidos a niveles superiores dentro de la pila de protocolos (por ejemplo para un protocolo de transporte como TCP o UDP) para futuros procesos. Estos niveles superiores pueden también generar datos sobre si mismos y pasarlos a niveles más bajos para encapsulación, enrutamiento y transmisiones eventuales.

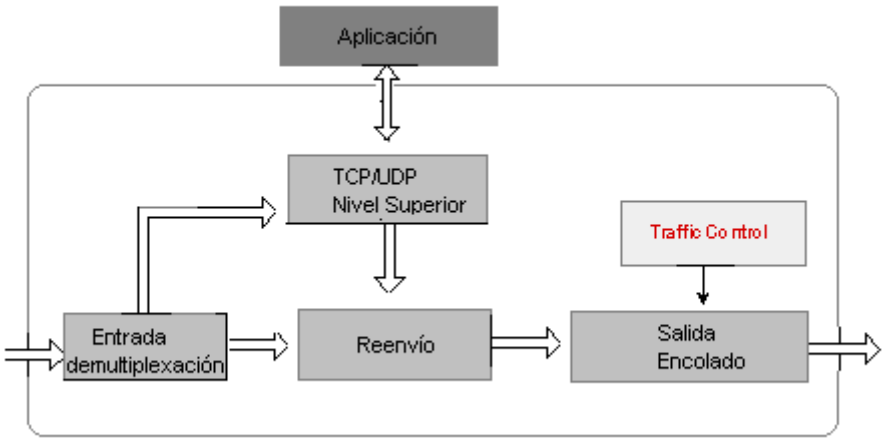


Figura 73 - Diffserv Linux

El reenvio incluye encapsulamiento, la selección del próximo salto, selección de la interfaz de salida etc. Una vez esto esta hecho, los paquetes son encolados sobe las interfaces respectivas. Este es el punto donde el control de tráfico entra a jugar. La disciplina de cola por defecto en linux tiene tres bandas y está basado en "pFIFO" (First in First Out). Cada vez que una interfaz es creada, la qdisc pfifo fast es automáticamente utilizada como una cola. En pFIFO, los paquetes pueden ser encolados en cualquiera de las tres bandas basados sobre sus bits Tipo de Servicio o asignados con prioridad.
Es posible revisar la cola pfifo fase y su longitud con el siguiente comando:
 # ip -6 addr show

Los paquetes que están a punto de ser enviados en la misma interfaz de red deben ser puestos primero en la cola y esperar para ser enviados. El encolado determina el orden en que los paquetes son enviados. Esto es importante dentro del mundo de internet ya que no se tiene un control sobre el tráfico UDP que es enviado. Para el tráfico TCP, se tiene muy limitado el control de congestión.
Una vez el control de tráfico ha decidido enviar un paquete, el equipo manejador lo recoge y lo emite sobre la red.

6.3.2.4 **Traffic Control (tc)**

Para configurar la regulación de tráfico en el kernel de linux se utiliza el comando *tc* el cual es parte del paquete iproute versión 2. Éste comando además de configurar las disciplinas de colas (qdisc), las clases (class) y los filtros (filter), permite también mostrar la configuración de las interfaces y estadísticas asociadas a éstas.
El comando ***tc*** se basa en lo siguiente para efectuar un control del tráfico:

- **Shaping:** Cuando se le da forma al tráfico, el índice de transmisión está bajo control. El "shaping" puede ser más bajo que el ancho de banda disponible, también se utiliza para suavizar las ráfagas repentinas de tráfico logrando un mejor comportamiento de la red. Es importante tener claro que el "shaping" ocurre en la salida (egress).

- **Scheduling:** Programando la transmisión de paquetes es posible mejorar la interactividad del tráfico que la necesite mientras se garantiza también el ancho de banda para las transferencias masivas, por otro lado, reordenar tiene el efecto de dar prioridades. Notar que el **scheduling** ocurre solo en la salida (egress).

- **Policing:** Mientras el "**shaping**" se ocupa de la transmisión del tráfico, el **policing** se ocupa de la llegada del tráfico, de esta forma se tiene que se aplica al ingreso (ingress).

- **Dropping:** El tráfico que excede el ancho de banda es eliminado tanto en la salida (egress) y la entrada (ingress).

El proceso del tráfico es controlado por tres tipos de objetos:

- **colas (qdisc):** Una disciplina de cola (qdisc) es un elemento importante bajo el concepto de control de tráfico, cada vez que el kernel desee enviar un paquete a la interfaz, éstos serán pasados a la disciplina de cola raíz (root) y ésta clasificará los paquetes en las colas respectivas mediante la invocación de sus filtros asociados o directamente como es el caso de la cola por defecto en la qdisc con clases HTB. Después, cada vez que el kernel

necesite paquetes para enviar a la interfaz, la qdisc raíz (root) será la encargada de pasar los flujos de datos a enviar.

- **Clases (class):** Una disciplina de colas con clases esta compuesta por varias colas internas denominadas colas hijas a las que se les denomina clases, así cuando se quiere encolar un paquete es la disciplina de colas la que efectúa el procedimiento. Por otro lado, cuando se quiere desencolar un paquete, la disciplina de cola de acuerdo a ciertas prioridades que pueden ser especificadas como parámetros con ***tc*** obtiene los paquetes de las colas hijas, controlando de esta forma la rapidez con se pasan los paquetes al kernel para ser enviados a la interfaz de red.

- **Filtros (filtres):** un filtro es usado por una disciplina de cola para determinar en que clase se encolará un paquete. Para que un flujo de bits llegue a una clase siempre debe ser enviado ahí por una qdisc, se recuerda nuevamente que los filtros son invocados sólo por las qdisc. Una disciplina de colas puede tener asociados varios filtros, se puede asignar prioridades para determinar que filtro se aplica primero, así los filtros se van aplicando uno a uno hasta que el paquete coincide con la configuración de un filtro y éste clasifica el paquete en una clase de la qdisc.

En la Tabla 22, se muestra una descripción del comando ***tc***, para cada tipo de objeto existen distintas operaciones: **add, change, replace y link**. Además se específica la interfaz de red que se interviene con ***tc*** mediante la sintaxis **dev DEV**, donde **DEV** puede ser *eth0*.

Para individualizar las instancias de los objetos se utiliza un ID, ésta identificación tiene una estructura compuesta por dos números enteros separados por el carácter ":", al primer entero se el denomina "número mayor" y al segundo "número menor", en el siguiente esquema se puede apreciar un ejemplo. En el caso de las qdisc, el número menor siempre es cero y una clase hija siempre hereda el número mayor de la qdisc padre (parent).

9 : 29

Número Mayor Número Menor

Estructura de los ID de los objetos

El ID para los objetos *qdisc* y *filters* se específica con el parámetro *handle*, mientras que para las clases se utiliza *classid***.**
El parámetro *parent* debe ser especificado para todos los objetos y corresponde al *handle* o classid ingresado para el objeto padre.
En el ítem 1 de la Tabla 22, se observa la sintaxis del comando ***tc*** para trabajar con disciplinas de colas (*qdisc*) y en el ítem 2 se utiliza para trabajar con clases (*class*), en ambos casos el parámetro qdisc que aparece al final indica la qdisc a utilizar con sus respectivos parámetros.

En el ítem 3 de la Tabla 22, se utiliza ***tc*** para trabajar con los filtros (filters). El parámetro *protocol* se utiliza para especificar el protocolo a utilizar, generalmente se trabaja con el protocolo IP, pero se pueden especificar otros. El parámetro *prio* se utiliza para especificar la prioridad del filtro, esto es importante cuando se tienen varios filtros pues permite indicar que filtro se ejecuta primero. El último parámetro, *flowid* corresponde al *classid* de la clase donde se va a clasificar el paquete, es decir, donde se va a encolar el paquete.

Los ítem 4, 5 y 6 muestran el uso de ***tc*** para mostrar estadísticas y datos de las instancias de los objetos asociados a una determinada interfaz de red.

1. ***tc*** qdisc [add | change | replace | link] dev DEV [parent qdisc-id | root] [handle qdisc-id] qdisc [qdisc specific parameters]
2. ***tc*** class [add | change | replace] dev DEV parent qdisc-id [classid classid] qdisc [qdisc specific parameters]
3. ***tc*** filter [add | change | replace] dev DEV [parent qdisc-id | root] protocol protocol prio priority filtertype [filtertype specific parameters] flowid flow-id
4. ***tc*** [-s | -d] qdisc show [dev DEV]
5. ***tc*** [-s | -d] class show dev DEV
6. ***tc*** filter show dev DEV

Tabla 22 Síntesis del comando tc.

Las operaciones que se pueden efectuar con los objetos son 5, sin embargo, como se aprecia en la Tabla 22 la operación *link* sólo se aplica al objeto *qdisc*, a continuación se describen las operaciones:

- **add:** agrega una disciplina de cola (*qdisc*), clase (*class*) o un filtro (*filter*) a una interfaz de red. Cuando se agrega una *qdisc* raíz se utiliza el parámetro *root*, para todos los demás objetos se debe indicar el parámetro *parent* (padre), el cual es el *handle* asignado a la *qdisc* de donde desciende.
- **remove:** elimina una instancia de un objeto. Para eliminar una clase se ingresa su *classid*. Para eliminar una qdisc se ingresa su *handle*, notar que cuando se elimina una disciplina de cola se elimina automáticamente todos los filtros asociados, todas las clases y qdisc descendientes, así, al eliminar la *qdisc* root (raíz) se elimina todo, inclusive las clases hojas. Si se quiere eliminar un filtro, se debe utilizar ***tc*** como se indica en el ítem 6 de la Tabla 22 para obtener el ID de la instancia.

- **change:** Algunos objetos pueden ser modificados sin ser movidos de su ubicación, compartiendo la sintaxis de la operación **add**, sin embargo, el *handle* no puede ser modificado ni tampoco el *parent*, es decir, un objeto no puede ser cambiado de posición.

- **replace:** Ésta operación es atómica en el sentido de que efectúa simultáneamente un **remove** y un **add** en un objeto. Si el objeto no existe entonces es creado.

- **link:** Ésta operación sólo puede ser utilizada en el objeto *qdisc* y realiza una sustitución donde el objeto debe existir.

Unidades de medida

Es muy importante conocer las unidades de medida con que trabaja el comando ***tc***, a continuación se muestra los diferentes tipos que utiliza.

Anchos de banda

kbps: Kilobytes por segundos
mbps: Megabytes por segundos
kbit: Kilobits por segundos
mbit: Megabits por segundos
bps o un número solo : Bytes por segundos

Cantidad de datos

kb or k: Kilobytes
mb or m: Megabytes
mbit: Megabits
kbit: Kilobits
b o un número solo: Bytes.

6.4 Desarrollo Técnico

Escenario1: Control de Tráfico utilizando flujos de video y paquetes de datos UDP

En el desarrollo de esta práctica se utilizarán seis PC's. Tres de ellos están actuando como enrutadores diffserv Ipv6 y son llamados A1, A2 y A3. Un pc llamado T es un generador de tráfico que inyecta tráfico a los enrutadores diffserv. Los otros dos pc actúan como transmisor y receptor; como se puede observar en la siguiente figura:

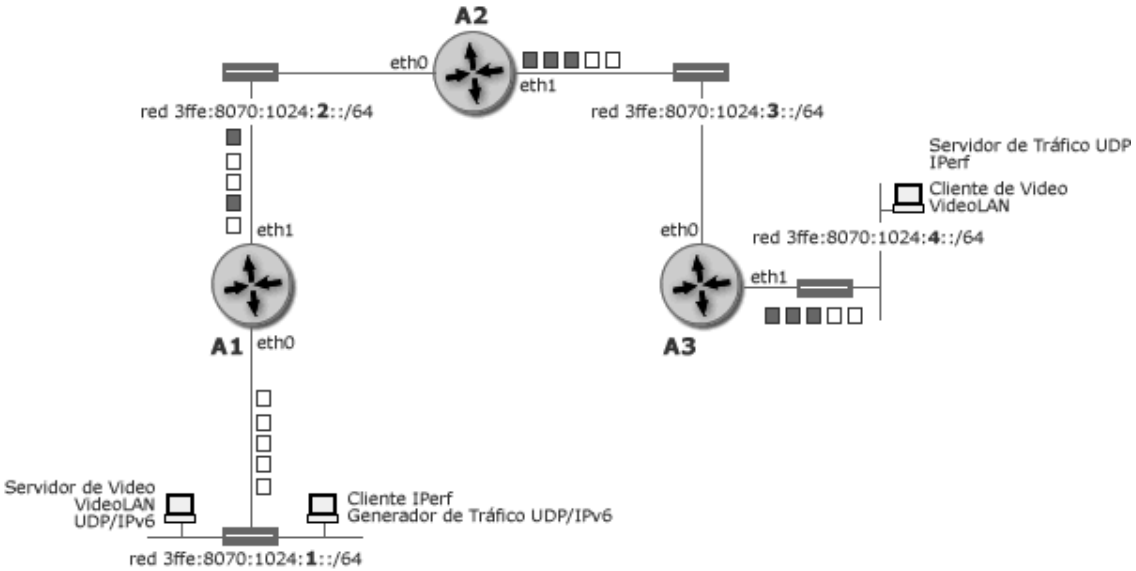


Figura 74 - Topología QoS Ipv6

Habilitar Soporte Diffserv:

Habilitar IPROUTE2:

El kernel de Linux a partir de la versión 2.2.X posee un subsistema de red totalmente rediseñado. La principal razón para esto fue que ante la necesidad de nuevas facilidades, que han ido surgiendo dentro del ámbito del networking, Linux fue añadiendo parches a su implementación de la pila de protocolos TCP/IP, lo cual provocó que el nivel de complejidad creciera enormemente haciéndolo prácticamente inmanejable.
La herramienta IPROUTE2 es la encargada de permitir configurar y gestionar todo este conjunto de nuevas facilidades, además de llevar a cabo todo el trabajo que hasta ahora se hacía con las clásicas herramientas ifconfig y route.
IPROUTE2 proporciona básicamente dos herramientas con las que implementar todas estas nuevas facilidades:

- La herramienta ip.
- La herramienta tc (traffic control) que es la que permitirá implementar toda la Gestión de tráfico.

Linux traffic Control está configurado con la herramiente TC. Esta hace parte del paquete iproute2. Algunas distribuciones de linux ya traen el soporte TC pero existen otras versiones viejas sin éste soporte. Si se presenta esta última opción es posible descargar la versión nueva del paquete iproute2 en este sitio ftp://ftp.proxad.net/mirrors/ftp.inr.ac.ru/ip-routing/. (*iproute2-2.4.7-now-ss020116-try.tar.gz*). Para habilitar el soporte tc se realiza el siguiente procedimiento:

Se extrae iproute2:

 # tar zxvf iproute2-2.4.7-now-ss020116-try.tar.gz

Se edita iproute2/Config para habilitar soporte Diffserv:

TC_CONFIG_DIFFSERV=y

Dentro de iproute2 se corre:

cd iproute2

make

Ahora tc se encuentra en :/sbin/tc

Las siguientes opciones de configuración del kernel tienen que estar habilitadas (enable) dentro de la sección Networking options.

Kernel/User netlink socket (CONFIG_NETLINK)

Network packet filtering (CONFIG_NETFILTER)

QoS and/or fair queueing (CONFIG_NET_SCHED)

Las siguientes opciones de configuración del kernel deben estar habilitadas en **Networking options**, **QoS and/or fair queueing**:

CBQ packet scheduler (CONFIG_NET_SCH_CBQ)

The simplest PRIO pseudoscheduler (CONFIG_NET_SCH_PRIO)

RED queue (CONFIG_NET_SCH_RED)

GRED queue (CONFIG_NET_SCH_GRED)

Diffserv field marker (CONFIG_NET_SCH_DSMARK)

Ingress Qdisc (CONFIG_NET_SCH_INGRESS)

QoS support (CONFIG_NET_QOS)

Packet classifier API (CONFIG_NET_CLS)

TC index classifier (CONFIG_NET_CLS_TCINDEX)

Firewall based classifier (CONFIG_NET_CLS_FW)

U32 classifier (CONFIG_NET_CLS_U32)

Traffic policing (CONFIG_NET_CLS_POLICE)

Luego de habilitarlas se pasa a recompilar el kernel asi:
make menuconfig
make dep; make bzImage; make modules; make modules_install;make install

Instalación y configuración de VIDEOLAN:

VideoLAN es una solución de software completa para transmisión de vídeo, desarrollada por estudiantes de Ecole Centrale. VideoLAN está diseñado para transmitir vídeo MPEG en redes con gran capacidad de ancho de banda.
La solución VideoLAN incluye:

- VLS (Servidor VideoLAN), el cual puede transmitir archivos MPEG-1, MPEG-2 y MPEG-4, DVDs, canales digitales de televisión terrestre y vídeo en vivo sobre la red en unicast o multicast,
- VLC (inicialmente cliente VideoLAN), el cual puede ser usado como servidor para transmitir archivos MPEG-1, MPEG-2 y MPEG-4, DVDs y vídeo en vivo sobre la red en unicast o multicast; o usado como cliente para recibir, decodificar y visualizar flujos MPEG sobre varios sistemas operativos.

Nota: antes de comenzar con la instalación de esta herramienta se presentan a continuación algunos comandos para manejar paquetes RPM dentro de linux redhat, los cuales serán necesarios para la configuración e instalación de las herramientas para enviar y recibir video:

Instalación
[root@ipv6]# Rmp –i <nombre del paquete>

Desinstalación
[root@ipv6]# Rmp –e <nombre del paquete>

Actualización
[root@ipv6]# Rmp –u <nombre del paquete>

Desplegar información a cerca de un paquete
[root@ipv6]# Rmp –qi <nombre del paquete>

Instalación de VideoLan cliente en Linux Redhat 9

Descargue e instale el paquete Kdelibs-3.1-10.i386.rpm que se encuentra dentro del paquete de Redhat
Si tiene la versión de mozilla1.2.1 es recomendable desinstalar galeon debido a que se podrían presentar inconvenientes al momento de instalar las librerías de videolan ya que este trabaja con mozilla1.4.7. para desinstalar galeon ejecute el siguiente comando:

[root@ipv6]# Rmp –e galeon-1.2.7-3

- Descargue los paquetes `vlc-binary.tar.gz` y `redhat9-updates.tar.gz` del enlace http://www.videolan.org/vlc/download-redhat.html y colóquelos dentro del mismo directorio.

Seguidamente descomprima e instale o actualice los paquetes RPM que ha descargado:
• descomprimir :

[root@ipv6 vlc]# ***tar zxf vlc-binary.tar.gz***
[root@ipv6 vlc]# ***tar zxf redhat9-updates.tar.gz***
[root@ipv6 vlc]# ***rpm -U *.rpm***

Si no ha instalado todos los paquetes RPM incluidos en su distribución, puede ser que se le pida instalar alguno de ellos previamente.

Instalación de VideoLan cliente en Windows

Descargue el paquete *vlc-0.7.1-win32.exe* ó el *vlc-0.7.1win32.zip* para bajar la carpeta o el archivo .zip y este último se extrae a una carpeta; desde el enlace www.vidolan.org/vlc/download-windows.html.

Transmitir sobre IPv6

Requerimientos
Se necesita un sistema operativo compatible con IPv6, como por ejemplo Linux 2.4 con el módulo *ipv6* cargado, Windows 2000 con la pila IPv6.

Transmitir con VLC

[root@ipv6 vlc]# ***vlc -vvv video1.xyz --ipv6 --sout udp:[direccion ipv6 del servidor]***

donde :

• video1.xyz es el fichero que quiere transmitir (también puede poner **dvdsimple:/dev/dvd** para transmitir un DVD, ...) ,
• *ff08::1* es : la dirección IPv6 de la máquina a la que quiere transmitir en unicast ; o la dirección IPv6 multicast .

Recibir sobre IPv6

Recibir un flujo unicast
[root@ipv6 vlc]# ***vlc -vvv --ipv6 udp:***

IPERF
Iperf es una herramienta para generar tráfico y medir el ancho de banda máxima de TCP, permitiendo medir de varios parámetros y de características de UDP. Iperf divulga anchura de banda, retrasa la inquietud y pérdida del datagrama.
Esta herramienta se puede descargar desde http://dast.nlanr.net/Projects/Iperf/#download.

Modo IPv6

Consiga la dirección IPv6 del nodo usando el comando del ' ifconfig.
Utilice - la opción de V para indicar que usted está utilizando una dirección IPv6.
-s Corre iperf en modo servidor.
-c Corre iperf en modo cliente conectado a un servidor iperf.
-u Para manejar tráfico UDP
-b Especifica el ancho de banda a enviar; por defecto es 1Mbit/sec.
-t Tiempo en segundos para transmitir. Por defecto son 10 seg
-i Fija un intérvalo de tiempo

Servidor:
[root@ipv6 vlc]# ./iperf - s - u - V

Cliente:
[root@ipv6 vlc]# ./iperf - c < dirección del servidor IPv6 > - u - V
nota: para conocer mas comandos de uso del iperf se recomienda el enlace: *http://dast.nlanr.net/Projects/Iperf/iperfdocs_1.7.0.html*

Configuración traffic control

Para el desarrollo de esta práctica se definieron dos scripts de configuración que se presentan a continuación utilizando la herramienta tc.
Para facilitar la ejecución de estos scripts, guarde esta configuración dentro de un archivo ubicado dentro de /usr/bin y modifique los permisos con chmod 755 <nombre del archivo> con el fin de hacerlos ejecutables desde cualquier parte.

Ejemplo:
[root@ipv6]*# chmod 755 qostc1_up*

qostc1_up: Enrutador de frontera

```
#! /bin/sh
TC=/sbin/tc
IFDEV=eth1

$TC qdisc add dev $IFDEV handle 1:0 root dsmark  indices 64
$TC class change dev $IFDEV classid 1:1 dsmark mask 0x3  value 0xb8
$TC class change dev $IFDEV classid 1:2 dsmark mask 0x3  value 0x0
$TC filter add dev $IFDEV parent 1:0  protocol ipv6  prio 4  u32  match ip6  src <Dirección IPv6 fuente> police rate 6Mbit  burst 10000K  flowid 1:1
$TC filter add dev $IFDEV parent 1:0  protocol ipv6  prio 6  u32  match ip6   src Dirección IPv6 fuente>police rate 500Kbit burst 10k  drop flowid 1:2
```

El script de configuración diffserv en la interfaz de frontera, implementa una cola dsmark para medir y remarcar sobre la cabecera del paquete DSCP diferenciando dos tipos de servicios el BE y el EF esta marcación se realiza con los valores mask 0x3 value 0x0 y mask 0x3 value 0xb8 respectivamente. En este script ser realiza una configuración de filtros dependiendo de la fuente que provenga especificando la dirección IPv6 de las fuentes.

qostc2_up: *Enrutador núcleo*

```
#! /bin/sh
TC=/sbin/tc
IFDEV=eth1

$TC qdisc add dev $IFDEV handle 1:0 root dsmark indices 64 set_tc_index
$TC filter add dev $IFDEV parent 1:0 protocol ipv6 prio 1 tcindex mask 0xfc  shift 2
```

```
$TC qdisc add dev $IFDEV parent 1:0 handle 2:0 cbq bandwidth 10Mbit allot 1514 cell 8 avpkt 1000 mpu 64

# Clase EF
$TC class add dev $IFDEV parent 2:0 classid 2:1 cbq bandwidth 10Mbit rate 6Mbit avpkt 1000 prio 1 bounded isolated allot 1514 weight 1 maxburst 100 defmap 1

$TC qdisc add dev $IFDEV parent 2:1 pfifo limit 5

$TC filter add dev $IFDEV parent 2:0 protocol ipv6 prio 1 handle 0x2e tcindex classid 2:1 pass_on

#Clase BE
$TC class add dev $IFDEV parent 2:0 classid 2:2 cbq bandwidth 10Mbit rate 500kbit avpkt 1000 prio 2 allot 1514 weight 1 maxburst 21 borrow

$TC qdisc add dev $IFDEV parent 2:2 red limit 60KB min 15KB max 45KB burst 20 avpkt 1000 bandwidth 10Mbit probability 0.4

$TC filter add dev $IFDEV parent 2:0 protocol ipv6 prio 2 handle 0x0 tcindex mask 0x0 classid 2:2 pass_on
```

Verificacion de configuración de Trafic Control

Una vez ejecutados estos scripts es posible comprobar su configuración de la siguiente forma:

[root@ipv6]# tc -s qdisc show dev eth1

También es posible observar con el siguiente comando entre otras cosas como se están clasificando y filtrando los paquetes. Además se puede apreciar las tasas actuales de transferencia, los paquetes fragmentados (giants), el ancho de banda prestado (borrowed) y otros datos concernientes al tipo de cola.

[root@ipv6]# tc -s class show dev eth1

[root@ipv6]# tc -s filter show dev eth1

Recursos:

Software

- 5 PC's con sistema operativo Red Hat Linux 9 (3 Enrutadores IPv6 2 Hosts IPv6)
- 1 PC con S.O Red Hat Linux o MS Windows 2000 (Hosts IPv6)
- Direcciónes IPv6 *unicast global agregable*, asignadas a las interfaz IPv6 de cada equipo enrutador
- 4 Hubs Ethernet
- Espacio de direcciones IPv6 (*bloque* pNLA /48): 3FFE:8070:1024::/48

Hardware

- Herramienta para transmitir video VideoLan.
- Herramienta generadora de tráfico Iperf

6.4.1 **Desarrollo de la Práctica**

Esta práctica se divide en dos fases:

- Fase 1: Control de tráfico en la red UniCauca IPv6 utilizando tráfico UDP
- Fase 2: Control de tráfico en la red UniCauca IPv6 utilizando tráfico UDP e ICMPv6

Fase 1: Control de tráfico en la red UniCauca IPv6 utilizando tráfico UDP

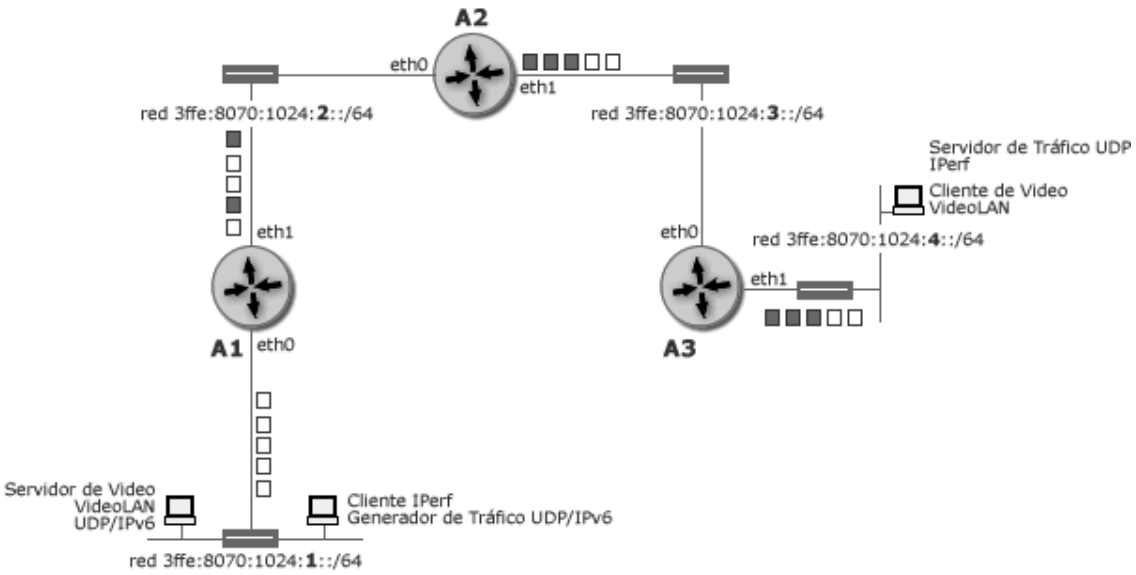


Figura 75 - Topología Escenario 1

En la Figura 75 se observa la topología a implementar en la fase 1, los enrutadores A1 y A2 son los que van a proveer la configuración de Control de Tráfico, el PC2 es el generador de tráfico, el PC3 es el equipo encargado de enviar flujos de video al equipo PC1.
En la siguiente tabla se especifica la asignación de direcciones a utilizar:

Enrutador	A1	A2	A3
[eth0]	3FFE:8070:1024:1::**1**/64	3FFE:8070:1024:2::**2**/64	3FFE:8070:1024:3::**3**/64
[eth1]	3FFE:8070:1024:2::**1**/64	3FFE:8070:1024:3::**2**/64	3FFE:8070:1024:4::**3**/64

Tabla 23 - Direcciones IPv6 de los Enrutadores

Procedimiento

Fase 1: Diffserv deshabilitado

- Configure los equipos A1, A2 y A3 con las direcciones que se presentan en la Tabla 23. Puede utilizar un protocolo de enrutamiento interno como *ospf* o *ripngd* vistos en las prácticas anteriores.

- En el equipo que Transmite el flujo de video (PC1) instale el VLC, descargue un video que maneje trafico UDP y configure el vlc para transmitir flujos Unicast.

- En el equipo Generador de tráfico configure el client *iperf* enviando paquetes de un tamaño constante 1032 (valor por defecto) de tipo udp y variando el ancho de banda para cada práctica según muestra la siguiente tabla:

Práctica	Ancho de Banda
1	1 Mbits/sec
2	2 Mbits/sec
3	3 Mbits/sec
4	4 Mbits/sec
5	5 Mbits/sec
6	7 Mbits/sec
7	10 Mbits/sec

- se utiliza el siguiente comando: # *./iperf - c < dirección IPv6 del servidor > - u -b x m -i 1 – V*, donde x indica los valores del ancho de banda a utilizar.

- En el equipo PC3 receptor de tráfico configure el servidor de vlc para recibir un flujo unicast ipv6 y el servidor de iperf para manejar tráfico udp.

- Ejecute el ethereal en el equipo PC3

- Envíe primero el tráfico de video y luego los paquetes UDP para cada caso especificado en la anterior tabla y observe la calidad del video y el reporte del servidor iperf.

P.6.1 ¿Que observa en el equipo receptor PC3 cada vez que se envían paquetes que consumen mayor ancho de banda de la red?

- Observe los resultados de ethereal relacionados con la secuencia de los paquetes de video y de tráfico.

P.6.2 ¿Cuáles son los valores de los campos Traffic Class y Flow Label para los dos tipos de paquetes?

- En PC2 ejecute los siguientes comandos y note los cambios efectuados en los campos Traffic Class y Flow Label.

  - *# ping6 -Q <0-7> <direccion IPv6 destino PC3>*
  - *# ping6 -F <0-f f f f f> <direccion IPv6 destino IPv6>*
  - *# ping6 -Q <0-7> -F <0 – f f f f f> <direccion IPv6 destino PC3>*

Fase 2: Diffserv habilitado

- Habilite Diffserv en los enrutadores A! y A2 ejecutando los scripts 1 y 2 respectivamente utilizando la configuración de los filtros según la fuente de donde provienen los paquetes y realice la verificación de configuración de traffic control.

- Ejecute en PC3 el ethereal.

- Envíe tráfico de video y tráfico de paquetes udp con diferente ancho de banda y observe los resultados del ethereal relacionados con la secuencia de los paquetes de video y de tráfico UDP.

P.6.3 ¿Qúe cambios observa en cada uno de los parámetros que entrega el reporte del servidor?

- Ejecute el ethereal en el equipo receptor de tráfico PC3.

P.6.4 ¿Observa cambios en el campo clase de tráfico dependiendo del tipo de paquetes? ¿explique la variación?

- En el script 1 equipo A1 modifique el ancho de banda asignado en el filtro u 32 con flowid 1:1 asignandole un valor de 500 K y al filtro u32 con flowid 1:2 un valor de 6 M, realice el mismo cambio en el script 2 equipo A2 en las clases cbq 2:1 y 2:2 respectivamente.

- En el equipo PC1 configure el cliente iperf al igual que en PC2 y envíe desde los dos equipos trafico iperf udp de 10 Mbits/sec.

P.6.5 ¿Explique el reporte de ancho de banda del servidor entregado a las dos fuentes?

- Repita el procedimiento de enviar video y tráfico iperf udp y oserve la calidad del video y el reporte del servidor.

P.6.6 ¿Cuáles son los cambios observados en la reproducción del video? ¿Por qué?

Escenario 2: Control de tráfico en la red UniCauca IPv6 utilizando tráfico UDP e ICMPv6

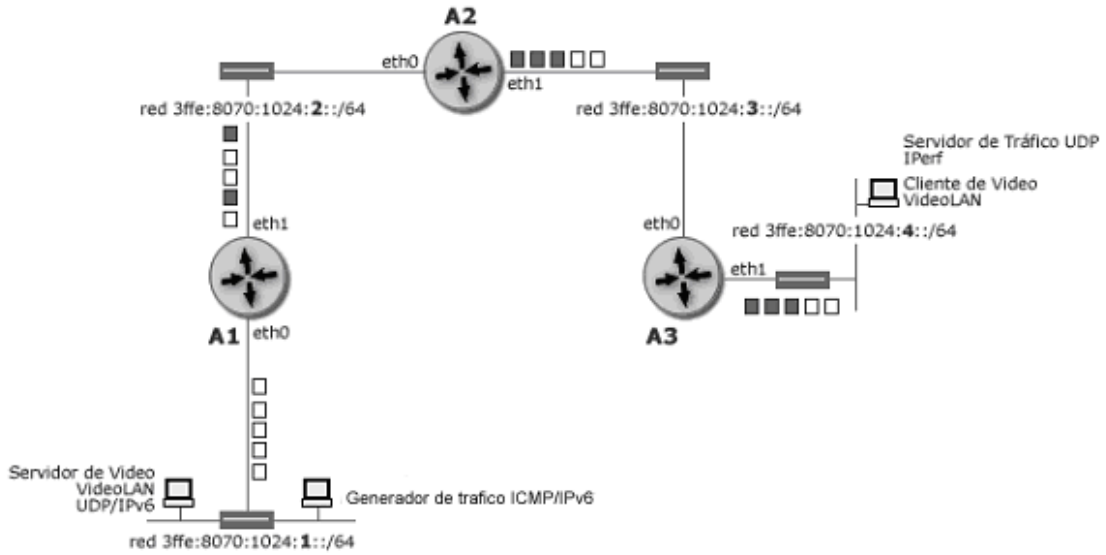


Figura 76 - Topología Escenario 2

Los enrutadores A1 y A2 son los que van a tener la configuración de control de tráfico, el PC3 es el receptor de tráfico, el equipo transmisor de flujos de video es el PC 1 y el generador de tráfico PC1. En la siguiente tabla se especifica la asignación de direcciones a utilizar:

Enrutador	A1	A2	A3
[eth0]	3FFE:8070:1024:1::**1**/64	3FFE:8070:1024:2::**2**/64	3FFE:8070:1024:3::**3**/64
[eth1]	3FFE:8070:1024:2::**1**/64	3FFE:8070:1024:3::**2**/64	3FFE:8070:1024:4::**3**/64

Tabla 24 - Direcciones IPv6

Procedimiento

Fase 1 Diffserv deshabilitado

- Configure los equipos A1, A2 y A3 con las direcciones que se presentan en la anterior tabla. Puede utilizar un protocolo de enrutamiento interno como *ospf* o *ripngd* vistos en las prácticas anteriores.
- En el equipo que Transmite el flujo de video (PC1) instale el VLC, descargue un video que maneje trafico UDP y configure el vlc para transmitir flujos Unicast.

- En el equipo Generador de tráfico PC2 envíe paquetes ICMPv6 variando los tamaños de los paquetes como se indica en la siguiente tabla y con intervalos de tiempo de 0.05 y 0.1, 0.3 y 0.5 para cada caso; el comando a utilizar es:

[root@ipv6]# *ping6 –s <tamaño de paquete icmp> -i <intervalo de tiempo> <Dirección IPv6 de destino PC3>*

Práctica	Tamaño de paquete ICMP (-s)
1	64 (valor por defecto)
2	1048
3	9000

- En el equipo PC3 receptor de tráfico configure el servidor de vlc para recibir un flujo unicast ipv6 y utilice ethereal para observar los paquetes icmp.

- Envíe primero el tráfico de video y luego los paquetes ICMP para cada caso especificado en la anterior tabla y observe la calidad del video y su relación con el tamaño y el intervalo de tiempo de los paquetes ICMPv6.

P.6.7 ¿Que diferencia se encuentra en el equipo receptor cada vez que se mandan paquetes de mayor tamaño y con menor tiempo?

P.6.8 ¿Qué puede concluir de lo realizado en los escenarios 1 y 2?

Fase 2: Diffserv habilitado

- Habilite Diffserv en los enrutadores A! y A2 ejecutando los scripts 1 y 2 respectivamente utilizando la configuración de los filtros según el tipo de protocolo del tráfico que se envía. Para esto modifique los filtros del script 1 equipo A1 de la siguiente forma:

```
$TC filter add dev $IFDEV parent 1:0  protocol ipv6  prio 4  u32  match ip6  protocol 17 0xff police rate 6Mbit  burst 10000K continue flowid 1:1

$TC filter add dev $IFDEV parent 1:0  protocol ipv6  prio 6  u32  match ip6  protocol 58 0xff police rate 5Kbit burst 10k  drop flowid 1:2
```

En la anterior tabla se observa la configuración de los filtros especificando los protocolos de los paquetes que se envían en este caso el video que utiliza el *protocolo UDP* y los ping6 que son *paquetes ICMP*. El número que se utiliza para determinar estos dos protocolos se encuentra dentro de /etc/protocols y el 0xff se utiliza para realizar una comparación de bits con el fin de comprobar que los paquetes que entran tienen este tipo de protocolo.

- Repita una vez más el anterior procedimiento enviando tráfico de video y paquetes ICMP con diferente tamaño de paquetes y diferentes intervalos de tiempo.

P.6.9 ¿Qúe cambios observa en cada caso?

- Ejecute el ethereal en el equipo receptor de tráfico

- ¿Observa cambios en el campo clase de tráfico dependiendo de que fuente o equipo provengan los paquetes? ¿explique la variación?

P.6.10 ¿Observe el campo de siguiente cabecera y la cabecera de fragmentación si es el caso, corresponde este valor a cada uno de los tipos de protocolos utilizados por cada paquete ?

- En el script 1 equipo A1 modifique el ancho de banda asignado en el filtro u 32 con flowid 1:2 un valor de 1.5K, realice el mismo cambio en el script 2 equipo A2 en las clase cbq 2:2.

P.6.11 ¿Explique el comportamiento tanto del video como de los paquetes ICMPv6?

6.4.2 **Análisis de Resultados**

6.4.3 **Conclusiones**

7 CONCLUSIONES Y RECOMENDACIONES

- El protocolo IPv6 fue diseñado teniendo en mente que el proceso de transición sería largo, y que tanto IPv4 como IPv6 tendrían que coexistir durante un amplio periodo de tiempo, así lo demuestra los grandes esfuerzos que realizan actualmente los diferentes grupos de trabajo en internet a nivel mundial, con el fin de mejorar las fallas que presenta IPv4 logrando con esto alargar su existencia.

- El registro del Sitio UniCauca IPv6 en 6bone y la implementación de la Red Piloto UniCauca IPv6 son el punto de partida en el largo proceso que deberá enfrentar UniCauca para apropiarse a todo nivel de ésta tecnología.

- Con el diseño y ejecución de las pruebas de protocolos de Enrutamiento IPv6 (RIPng,OSPFv6 y BGP4+), de Seguridad (IPSec) y de Calidad de servicio sobre la Red piloto Unicauca IPv6, se logró obtener un laboratorio Internet IPv6 que permitirá a la comunidad estudiantil de la FIET interactuar con estos protocolos, conociendo a fondo su funcionamiento y el papel que cada uno tiene dentro del correcto desempeño de una red basada en IPv6.

- Se recomienda continuar con la investigación e implementacion de este protocolo dentro de la Universidad del Cauca, con el fin de aprovechar las ventajas que tiene a nivel de aplicaciones y la oportunidad de entrar a formar parte de Internet 2.

BIBLIOGRAFÍA

- Bieringer P. (1998). *Peter Bieringer's IPv6 & Linux*. www.bieringer.de/linux/IPv6.
- Conta A., Deering S. (1998). *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6)*, RFC 2463.
- Gilligan R., Nordmark E. (1996). Transition Mechanisms for IPv6 Hosts and Routers, *RFC 1933*.
- Bert Hubert (2000). Enrutamiento Avanzado y control de Tráfico en linux. http://www.gulic.org/comos/LARTC/lartc.html
- RFC2460, Especificaciones del Protocolo Internet Versión 6 (IPv6)
- RFC2461, Descubrimiento del Vecindario para IPv6 (ND)
- RFC2462, Autoconfiguración de Direcciones "stateless" IPv6
- RFC2463, Protocolo de Mensajes de Control de Internet para IPv6 (ICMPv6)
- RFC1981, Descubrimiento del MTU de la ruta para IPv6
- RFC2373, Arquitectura de Direccionamiento en IPv6
- RFC1887, Arquitectura para la Asignación de Direcciones Unicast IPv6
- RFC2374, Formato de Direcciones Unicast Agregables Globales
- RFC2283, Extensiones Multiprotocolo para BGP-4
- RFC2545, Uso de las Extensiones Multiprotocolo de BGP-4 para Routing entre dominios para IPv6
- RFC2740, OSPF para IPv6
- RFC2402, Cabecera de Autenticación IP
- RFC2406, Encriptación de datos en IP (ESP)
- RFC2185, Aspectos de Routing de la Transición IPv6
- RFC2473, Especificaciones Genéricas de Tunelización de Paquetes en IPv6
- RFC2529, Transmisión de IPv6 sobre Dominios IPv4 sin Túneles Explícitos

Referencias Web

- IETF: **http://www.ietf.org/html.charters/ipv6-charter.html**
- 6BONE: **http://www.6bone.net**
- 6TAP exchange: **http://www.6tap.net**
- Formato de Registro al 6Bone: **http://www.viagenie.qc.ca/en/ipv6/registry/newusers.shtml**
- IPv6 Forum: **http://www.ipv6forum.com**

ACRÓNIMOS

A

ABR, Area Border Router

AH , Authentication Header

ALE, Address Lifetime Expectations

AAL, ATM adaptation layer

AP-NIC, Asia Pacific - NIC

ARIN: American Registry for Internet Numbers

AS, Autonomous System

B

BGP4+, Border Gateway Multicast Protocol to IPv6

C

CATNIP, Common Architecture for the Internet

CBQ, Class-Based Queueing

D

DSCP, Diffserv Code Point

E

EF, Expedited Forwarding PHB

EGP, Exterior gateway protocol

ESP, IP Encapsulating Security Payload

EIGRP - Enhanced interior gateway routing protoco

F

FIFO, first in first out

FDDI, Fiber Distributed Data Interface

FO, Fibra Óptica

G

GRE Generic Routing Encapsulation

H

HTB ,Hierarchical Token Bucket

I

IANA, Internet Assigned Number Authority

IETF, Grupo de trabajo de Ingeniería en Internet

ICMPv4, Internet control message protocol

ICMPv6, Internet control message protocol

IDRP, Interdomain routing protocol

IGMPv4, Internet group management protocol

L

LAN, Local Area Network

M

MLD (multicast listener discovery

N

NLA ID, Next Level Aggregation Identifier

ND, Neighbor Discovery

O

OSPF,Open shortest path first

OSI, Open System Interconnection

OSPF, Open Shortest Path First

Q

QoS, Quality of Service

R

RFC, Request For Comment

RIP, Routing Information Protocol

RIPE-NIC: Reseaux IP Europeennes - NIC

S

SIPP,Simple Internet Protocol Plus

SLA ID, Site Level Aggregation Identifier

T

TCP, Transport Control Protocol

TTL, Time To Live

TUBA ("TCP and UDP with Bigger Addresses

TLA ID, Top Level Aggregation Identifier

U

UDP, User Datagram Protocol

UNAM, Universidad Nacional Autónoma de México

USAGI, UniverSAl playGround for Ipv6

V

VLC, VideoLan Client