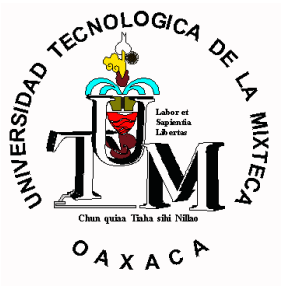

Universidad Tecnológica de la Mixteca

IPv6: El nuevo protocolo de Internet, instalación, configuración, pruebas y análisis de paquetes en Linux

TESIS

PARA OBTENER EL TÍTULO PROFESIONAL DE:

INGENIERO EN COMPUTACIÓN

PRESENTA:

Carlos Hiram Hernández Sánchez

ASESOR:

M.C. GABRIEL GERÓNIMO CASTILLO

HUAJUAPAN DE LEÓN, OAXACA ABRIL DE 2002

Lista de figuras

Capítulo 4

Capitulo 5

Lista de tablas

Índice General

Introducción

El explosivo crecimiento de computadoras, y ahora dispositivos móviles, conectadas a Internet en la pasada década y en los comienzos de esta, hace que el protocolo IP tenga que renovarse o morir, esto ha sido previsto por la IEFT[1] desde principios de la década pasada, el resultado final, a mediados de los 90, se le conoce como IPv6 (Internet Protocol version 6).

IPv6 comprende una serie de mejoras, las direcciones en IPv4 son de 32 bits, en IPv6 son de 128 bits, lo que proporciona un espacio "inagotable"[2] de direcciones para delegar a los usuarios de Internet, maneja la seguridad y la encriptación de manera intrínseca, la calidad de servicio – tan necesaria para transmisiones de video o voz sobre Internet – etc.

Los objetivos de este trabajo de tesis son los siguientes:

- Sentar las bases para que se inicie la investigación sobre IPv6 en la Universidad Tecnológica de la Mixteca.
- Configurar un host que sea capaz de manejar direcciones de IPv6.
- Construir un Sniffer capaz de capturar y visualizar paquetes IPv6.

Para lograr los objetivos se han realizado los siguientes capítulos. En el primero se trata sobre IPv4, donde se habla un poco de su historia, generalidades y su problemática actual.

En el segundo capítulo se habla un poco del sistema operativo – Linux - seleccionado sobre el que corre el servidor configurado en esta tesis, también se habla de las licencias para el Software Libre.

En el tercer capítulo, se habla acerca de IPv6, características, mejoras, protocolos de extensión y estructuras de datos que los contienen.

[1] *Internet Engineering Task Force*
[2] Esto puede ser sometido a discusión, ya que ideas similares se tuvieron a principios de los 80 cuando fue liberado IPv4.

En el cuarto capítulo se habla de la implementación en Linux para dos diferentes versiones del sistema operativo, corriendo dos diferentes generaciones de Kernel, así como de la actualización de paquetes mínimos necesarios para el funcionamiento del servidor.

En el quinto capitulo se habla de un sniffer realizado para comprobar los paquetes que circulan en la red funcionando con IPv6, se capturan y se analizan utilizando las estructuras de datos definidas por el estándar POSIX.

En el sexto capítulo se habla del trabajo a futuro de esta tesis, y la repercusión que puede tener sobre la red universitaria para una planeación a futuro, la creación de empresas de consultoría, etc.

Finalmente se habla de las conclusiones, de las que se puede adelantar la necesidad de aprender y experimentar con este nuevo protocolo y no esperar ser sólo consumidores de lo que se investiga y desarrolla en otras instituciones o empresas.

Capítulo 1. IPv4

Introducción.

Antes de proceder con un repaso en lo referente a IPv4 se describirá brevemente lo que es Internet, el más grande campo de investigación y al que le deben su creación los protocolos de comunicación de datos en redes sobre los que gira este trabajo.

1.1. Una breve historia de Internet.

Los orígenes de lo que hoy conocemos como Internet se remontan al final de la década de 1960, específicamente a 1969, año en el que el Departamento de la Defensa (DARPA) de los Estados Unidos trabajaba en un proyecto denominado ARPANET, el objetivo principal del proyecto era contar con una red de comunicaciones fiable, capaz de comunicar puntos distantes entre sí en caso de que la infraestructura telefónica se viera afectada por algún tipo de ataque militar. A finales de ese mismo año se logro la conexión de 4 hosts en una red inicial, la cual fue creciendo y expandiéndose rápidamente, en los años subsecuentes se empezó a investigar la manera de transmitir información entre diferentes redes no necesariamente compatibles, así fue como se consiguió enlazar computadoras de redes independientes comunicándose de forma transparente. Este proyecto recibió el nombre de 'Internetting', y el sistema de redes funcionando conjuntamente formando una red mayor se la denominó 'Internet'.[1]

Con el paso del tiempo las funciones militares se separaron del ámbito de desarrollo permitiendo así el acceso a la red a todas aquellas instituciones que lo requirieran, siempre y cuando dicho acceso fuera para fines académicos o de investigación. A finales del año 1972 ya había mas de 40 nodos conectados a la red, entre ellos, caben destacar las de las universidades de California en Los Ángeles (UCLA), el Instituto de Investigaciones de Stanford (RSI), la Universidad de California Santa Bárbara (UCBA), etc.

En el año de 1992 dejó de funcionar la red que dio origen a Internet, ARPAnet, pero en ese mismo año, Tim Berners Lee (actualmente presidente del

[1] La transmisión de la información en la red. A la interconexión de redes, una resultante "meta-red" se la denomina Internet. Obsérvese aquí la sutil diferencia entre *una* Internet y *la* Internet. El último es el nombre oficial de una Internet global en particular.

WWW Consortium) creó el lenguaje HTML (Hyper Text Markup Language) con el que nació la World Wide Web.[61]

El fenómeno de compartir información rápidamente atrajo la atención de universidades y oficinas gubernamentales así como de redes privadas que querían conectarse a está nueva red. El crecimiento continuo durante la década de 1980, pero no fue hasta la década pasada cuando el proyecto se convirtió en un nuevo medio de comunicación y porque no, una nueva forma de vida, esto en gran medida al desarrollo de medios que hicieron el uso de Internet más fácil y agradable permitiendo así, que gente con escasa preparación en computación se adentrara en territorios que antes pertenecían a ingenieros, propiciando con esto su popularidad y aceptación en el mundo entero.

Una muestra de esto es que en 1991 había unas 535.000 computadoras conectadas a Internet, de las cuales el 48% estaba dedicada actividades docentes contra un 34% dedicadas a actividades comerciales; a finales de 1996 un 45% de los equipos conectados a Internet tenia propósito comercial. [61]

1.2. Generalidades.

El protocolo IP fue diseñado para interconectar sistemas basados en redes de intercambio de paquetes. Dicho modelo se ha venido llamando catanet. Este protocolo permite el intercambio de bloques de datos entre hosts[2], denominados datagramas, conectados a una red, cada uno de ellos tiene una dirección única denominada dirección IP.

IP implementa dos funciones básicas: el encaminamiento y la fragmentación. Para la primera de las funciones se sirve de uno de los campos que aparecen en la cabecera de los datagramas, se trata de la dirección del host destino. Esta dirección es utilizada para transmitir los datagramas hacia el host correspondiente. La segunda de las funciones está influenciada por el nivel en que se encuentra situado justo debajo de la capa IP, se trata del nivel de enlace. Los datagramas generados por IP deben amoldarse al tamaño máximo que es capaz de tratar la red, el cual está limitado por la capa de nivel de enlace. Si el tamaño de una trama es menor que el datagrama generado por IP, entonces la capa IP se ve obligada a fragmentar, de manera que los datagramas resultantes pueden ser enviados por la red.

El protocolo IP trata cada datagrama como una unidad independiente del resto de los datagramas, no existen conexiones ni circuitos virtuales en el envío de la información.

El IETF publicó las especificaciones para IPv4 en el [RFC791] en el otoño de 1981. Cuando las especificaciones del IPv4 fueron liberadas, Internet era una

[2] Un host es un dispositivo que puede recibir y brindar servicios de red.

comunidad de aproximadamente 100 sistemas. Las especificaciones de IPv4 dicen que todas las direcciones IP deben ser representadas por un número de 32 bits formado por 4 grupos de números de 8 bits. Esto proporciona un total de algo mas de 4 millones de direcciones, sin embargo solo unos pocos cientos de miles están actualmente disponibles debido a los esquemas de asignación jerárquica.

Desde la liberación de IPv4, la población de Internet ha crecido a más de 100 millones de computadoras, incrementándose más rápido que lo anticipado, como el total de direcciones disponibles decrece, será muy difícil obtener direcciones IPv4. Más aun, este paso de crecimiento se espera continué los próximos años. [46]

Las primeras asignaciones de direcciones IP dieron direcciones para algunas corporaciones e instituciones en bloques muy grandes. Esas asignaciones de redes clase A y clase B fueron decretadas cuando el crecimiento de Internet no estaba previsto. Mientras algunos de esos usuarios primarios aún tienen direcciones disponibles para el uso interno, el total de direcciones no usadas está llegando a ser más pequeño cada día. Esas direcciones que fueron repartidas a algunas compañías para sus grandes redes corporativas no pueden ahora ser reutilizadas por otros.

Una dirección IP está representada por cuatro números enteros, cada uno de ellos de un byte y separados por un punto. Las direcciones IP se componen de dos partes, la primera de ellas hace referencia a una red y la segunda a un host concreto dentro de la red. Dependiendo del número de bytes que se destine a cada parte se pueden encontrar cuatro tipos de redes:

- **Redes clase A**

  Destinan un byte para identificar la red y tres bytes para identificar los host dentro de dicha red.

RED-ID	HOSTS-ID
0	

  El bit más significativo del byte destinado a la red tiene el valor de cero, por lo que los rangos de redes posibles van desde 1 hasta 127. Mediante los tres bytes destinados a los host se direccionan mas de 16 millones de hosts en cada red.

  Ejemplo: 12.100.20.30

  que representa la red 12 y el host 100.20.30 dentro de dicha red.

- **Redes de clase B**

Destinan dos bytes para identificar la red y dos bytes para identificar a los hosts dentro de ella.

RED-ID	HOSTS-ID
10	

Los dos bits más significativos de los destinados a identificar la red, tienen el valor uno y cero respectivamente. Esto implica que los rangos de redes permitidos van desde la 128.1 hasta la 191.254, esto es, hay 16382 posibles redes clase B y mediante los dos bytes destinados a los hosts se pueden representar 65534 hosts en cada red.

Ejemplo: 141.17.90.239

que representa la red 141.17 y el host 90.239 dentro de ella.

- **Redes clase C**

Destinan tres bytes para identificar la red y un byte para identificar a los hosts

RED-ID	HOSTS-ID
110	

Los tres bits más significativos de los bytes que identifican la red valen uno, uno y cero respectivamente. Esto permite rangos que van desde 192.1.1 hasta 223.254.254, esto es, hay 4194304 redes clase C y mediante el byte destinado a los hosts se puede identificar a 254 equipos.

- **Redes de clase D**

Son redes cuya dirección IP es especial, ya que está reservada para grupos de multienvío. Su primer byte puede valer cualquier número entre 224 y 239, estos números identifican la red, los bytes restantes pueden identificar el número de multienvío.

En todos los casos anteriores se puede apreciar que tanto el número de redes como el número de hosts que realmente se pueden direccionar con los bytes utilizados es mayor que el número citado anteriormente. Esto es debido a que existe una serie de direcciones que están reservadas para propósitos especiales. Por ejemplo, en una red clase C se podrían direccionar 256 equipos (2^8), sin embargo, la dirección 0 (00000000 en binario) y la 255 (11111111) están reservadas, por lo que el número total de equipos que se pueden direccionar es de

254, la dirección 0 esta reservada para identificar la red mientras que la 255 lo está para envíos en forma de broadcast.[3]

Los esfuerzos para explotar al máximo las direcciones con que aun se cuenta, han dado como resultado algunas técnicas, tales como las mascaras de subred, las direcciones sin clase, subredes (subnetting), NAT (network address translation), etc.

1.3. Formato del encabezado de IPv4.

Dos de las características principales del protocolo IP son que se trata de un protocolo no orientado a conexión y no fiable. Al ser no orientado a conexión, realiza la comunicación a través de paquetes, llamados datagramas. Al no existir conexión ni realizarse circuitos virtuales, los datagramas son tratados con encaminamiento individual. Esto significa que distintos datagramas procedentes de un mismo origen y con un mismo destino pueden llegar en desorden, duplicados o simplemente, perderse. Esto es lo que en arquitecturas de protocolos se entiende por protocolo no fiable.

Se trata de un protocolo best effort[4] (lo hará lo mejor que pueda), lo que significa que, pese a que es admisible que descarte datagramas, no lo hará caprichosamente.

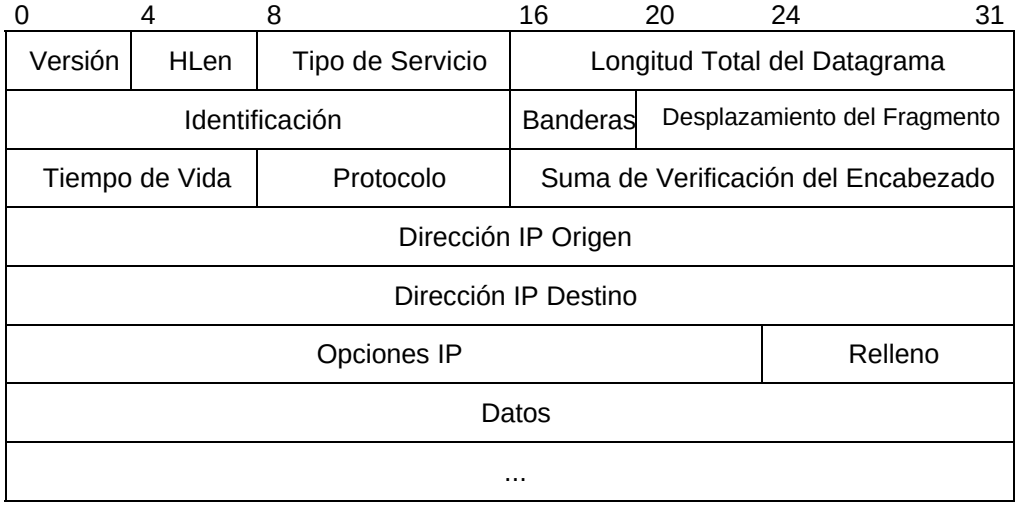

Figura 1. Estructura del datagrama IPv4

La figura 1 muestra el formato de un datagrama IPv4. Como es habitual en cualquier protocolo, consta de dos partes principales: una cabecera con información necesaria para el protocolo, y un cuerpo con los datos transportados. La cabecera consta de una serie de campos obligatorios, lo que hace que su

[3] Envío de información a todos los hosts en la red.
[4] Traducción literal, Mejor Esfuerzo.

longitud mínima sea de 20 bytes, y los otros opcionales. El significado y función de los mismos es el siguiente:

- Versión: 4 bits. Para IPv4 vale siempre 4.

- HLen: Longitud de la cabecera: 4 bits. Como se ha dicho, la cabecera IPv4 tiene longitud variable, en función de los campos opcionales que utilice. Por ello se hace necesario este campo, con objeto de conocer dónde empieza el cuerpo de los datos.

- Tipo de Servicio: 8 bits. Este campo es un intento (fallido) de diferenciar calidades de servicio. Está compuesto por un subcampo que indica la prioridad del datagrama y flags (banderas) de un bit para indicar a los Routers[5] (ruteadores) cuál es la preferencia del mismo si debe elegir entre retardo, caudal o fiabilidad. Sin embargo, desde las primeras implementaciones se ha hecho caso omiso de este campo, por lo que se puede afirmar que no tiene uso.

- Longitud Total del Datagrama: 16 bits. Indica la longitud en octetos de todo el datagrama, incluyendo la cabecera. Por tanto, la longitud máxima de un datagrama IPv4 será: $2^{16} = 64Kb$

- Campos de fragmentación: 16 bits.

- Tiempo de vida: 8 bits. (TTL: Time To Live) del datagrama. Este campo expresaba en su definición original un tiempo de vida de los datagramas. Transcurrido éste, se descartaría. El objeto de esto sería defenderse ante bucles en el encaminamiento. En la práctica, se ha empleado como un contador decreciente de saltos, un valor que decrementará en una unidad en cada router que atraviesa el datagrama, con la misma finalidad. Un uso posterior que se le ha dado es limitar la distancia en saltos de las difusiones multicast.

- Protocolo: 8 bits. Indica a qué protocolo de nivel superior corresponden los datos transportados.

- Suma de Verificación del Encabezado: 16 bits. Contiene información redundante para comprobar la integridad de los datos de la cabecera.

- Dirección IP Origen: 32 bits. Dirección IPv4 del remitente del datagrama.

- Dirección IP Destino: 32 bits. Dirección IPv4 del destinatario último del datagrama.

[5] Router, encaminador, dispositivo que conecta cualquier número de redes de área local.

- Campos Opcionales: A continuación de la parte fija de la cabecera y antes del cuerpo de datos, cabe la posibilidad de insertar campos opcionales como encaminamiento en origen, registros de ruta (anotación de los nodos intermedios por los que pasa el datagrama) o timestamping[6] (posibilidad de que cada nodo intermedio marque en el datagrama, la hora en que éste pasa por él).

1.4. Problemática de IPv4

Si todas las personas en el mundo tuvieran una conexión a Internet, podríamos no tener suficientes direcciones IPv4 para todos. Es aun posible que en el futuro cada persona tenga muchas direcciones IP, una casa inteligente puede tener direcciones IP para cada dispositivo, por ejemplo, un refrigerador que cuando sepa que falta algo pueda hacer un pedido a un supermercado inteligente, o para disminuir el nivel de las luces en una casa. Esto puede requerir tantas como 800 – 1,000 billones de direcciones IP. Una muestra de esto son los teléfonos celulares, los pagers, las agendas que permiten enviar y recibir correo electrónico, palms, etc., son dispositivos que probablemente requieran en un futuro una dirección IP homologada.

Otra vertiente por explorar es la conformación del proyecto Internet 2, el cual es muy probable que ocupe IPv6.
A continuación se mencionan algunos ideas de porque el cambio de protocolo deber ser inminente

Escala:

Cada máquina presente en la red dispone de una dirección IP de 32 bits. Ello supone más de cuatro mil millones de máquinas diferentes. Esa cifra, no obstante, es muy engañosa. El número asignado a una computadora no es arbitrario, sino que depende de una estructura más o menos jerárquica (en especial, pertenece a una red), lo cual ocasiona que se desperdicie una enorme cantidad de direcciones. La cuestión es que en 1993 fue claro que con el ritmo de crecimiento sostenido de Internet hasta aquel momento (exponencial), el agotamiento del espacio de direcciones era casi inminente.

Enrutamiento:

Otro de los grandes problemas del crecimiento de Internet es la capacidad de almacenamiento necesaria en los routers y el tráfico de gestión preciso para mantener sus tablas de ruteo. Existe un límite tecnológico al número de rutas que un nodo puede manejar, y como dado que Internet crece mucho más rápidamente que la tecnología que la mantiene, se vio que los gateway[7] pronto alcanzarían su

[6] Etiquetas de Tiempo.
[7] Puerta en comunicaciones de red, un nodo en una red que sirve como una entrada a otra red. Pasarela.

capacidad máxima y empezarían a desechar rutas, con lo que la red comenzaría a fragmentarse en subredes sin acceso entre sí.

Dado lo grave de la situación se definió el CIDR[8] (Classless Inter-Domain Routing) [RFC1481] [RFC1517] [RFC 1518] [RFC 1519], con el que las pasarelas reducían el tamaño de sus tablas segmentando una red en varias subredes. Gracias a ello se ha ganado un tiempo muy valioso, pero tan sólo se ha postergado lo inevitable.

En [RFC1797] y [RFC1879] se realiza el experimento de dividir una red A (la red 39) en multitud de pequeñas subredes. Los resultados fueron alentadores, por lo que dicha técnica podría utilizarse para ampliar de nuevo el tiempo de vida de IPv4.

Multiprotocolo:

Cada vez resulta más necesaria la convivencia de diversas familias de protocolos: IP, OSI, IPX... Se necesitan mecanismos que permitan abstraer al usuario de la tecnología subyacente para permitir que concentre su atención en los aspectos realmente importantes de su trabajo. Se tiende, pues, hacia una red orientada a aplicaciones, que es con lo que el usuario interacciona, más que a una red orientada a protocolos (como hasta el momento) [RFC1560].

Seguridad:

El mundo IPv4 es el mundo académico, científico, técnico y de investigación. Un ambiente, en general, que podría calificarse como "amigable", desde el punto de vista de la gestión y la seguridad en la red. Con la aparición de servicios comerciales y la conexión de numerosísimas empresas, el enorme incremento en el número de usuarios y su distribución por todo el planeta, y la cantidad, cada vez mayor, de sistemas que necesitan de Internet para su correcto funcionamiento, es urgente definir unos mecanismos de seguridad a nivel de red. Son necesarios esquemas de autentificación y privacidad, tanto para proteger a los usuarios en sí como la misma integridad de la red ante ataques malintencionados o provocados por errores [RFC1281] [RFC1636] [RFC1828] [RFC1829].

Tiempo Real:

[8] CIDR es una abreviatura de Enrutamiento Inter-dominio sin Clases. Como se mencionó, las direcciones IP se asignaban por clases, con el advenimiento de CIDR, el espacio de direccionamiento se asigna en fronteras de bits. Al usar CIDR es posible asignar el espacio de direcciones que se corresponda con el número de servidores en su red, ahorrando por tanto espacio de direcciones, es decir, solo se asignan las necesarias.

IPv4 define una red pura orientada a datagramas y, como tal, no existe el concepto de reserva de recursos. Cada datagrama debe competir con los demás y el tiempo de tránsito en la red es muy variable y sujeto a congestión. A pesar de que en la cabecera IP hay un campo destinado a fijar, entre otras cosas, la prioridad del datagrama [RFC1349] [RFC1455], en la práctica ello no supone ninguna garantía. Se necesita una extensión que posibilite el envío de tráfico de tiempo real, y así poder hacer frente a las nuevas demandas en este campo [RFC1667].

Tarificación:

Con una red cada día más orientada hacia el mundo comercial hace falta dotar al sistema de mecanismos que posibiliten el análisis detallado del tráfico, tanto por motivos de facturación como para poder dimensionar los recursos de forma apropiada [RFC1272] [RFC1672].

Comunicaciones Móviles:

El campo de las comunicaciones móviles está en auge, y aún lo estará más en un futuro inmediato. Se necesita una nueva arquitectura con mayor flexibilidad topológica, capaz de afrontar el reto que supone la movilidad de sus usuarios. La seguridad de las comunicaciones, en este tipo de sistemas, se ve, además, especialmente comprometida [RFC1674] [RFC1688].

Facilidad de Gestión:

Con el volumen actual de usuarios y su crecimiento estimado, resulta más que obvio que la gestión de la red va a ser una tarea ardua. Es preciso que la nueva arquitectura facilite al máximo esta tarea. Un ejemplo de ello sería la auto configuración de los equipos al conectarlos a la red [RFC1541].

Política de enrutamiento:

Tradicionalmente los datagramas se han encaminado atendiendo a criterios técnicos tales como el minimizar el número de saltos a efectuar, el tiempo de permanencia en la red, etc. Cuando la red pertenece a una única organización eso es lo ideal, pero en el nuevo entorno económico en el que diferentes proveedores compiten por el mercado las cosas no son tan simples. Es imprescindible que la fuente pueda definir a través de cuales redes desea que pasen sus datagramas, atendiendo a criterios de fiabilidad, coste, retardo, privacidad, etc. [RFC1674] [RFC1675].

Capítulo 2. Documentación del sistema operativo

Introducción.

Ya que el sistema operativo es el administrador de los recursos con que cuenta una computadora, y en algunos casos de toda una red, resalta la importancia que reviste la adecuada selección del Sistema Operativo con el que se trabajará, por ello se da una justificación a la selección del sistema operativo, GNU Linux Red Hat [52] y una breve historia de este.

2.1. Justificación.

Como se mencionó, el sistema operativo es el administrador de los recursos de hardware y software de la máquina, además de presentar una interfaz amigable para el usuario. Una computadora sin un sistema operativo es una caja inútil de plástico y componentes electrónicos que no hacen nada, la computadora está inactiva, incapaz de responder a los comandos o peticiones que se le hagan.

La mayoría de los sistemas operativos son software propietario, soportados por grandes compañías de software, al adquirir uno de esos sistemas operativos debe conformarse con lo que le ofrece el proveedor, entonces no se puede modificar el sistema ni experimentar con él.

Linux es de los pocos sistemas operativos libremente disponibles que soportan multitarea y multiproceso para múltiples usuarios[1]. Linux ofrece estabilidad y potencia, además de ser ajeno a los caprichos de marketing de las empresas comerciales, no es obligatorio actualizarse cada cierto tiempo (aunque por cuestiones de seguridad se hace una obligación) ni pagar por las actualizaciones, es más, muchas aplicaciones están disponibles en Internet listas para descargarse y usarse.

La diferencia que existe con los Sistemas Operativos comerciales es que el usuario tiene que aceptar lo que le vende, en Linux el usuario tiene acceso al código fuente para modificar y ampliar el sistema y adaptarlo a sus necesidades.

[1] Cabe mencionar que existen otros sistemas operativos como OpenBSD o FreeBSD, los cuales cumplen el estándar POSIX y están bajo la licencia GNU.

Linux en si es libre, el empaquetado y los programas adicionales son los que tienen un costo, siempre y cuando se incluya el código fuente.

Linux ofrece la posibilidad de probar y aprender, todo el código fuente de un sistema operativo funcional y completo está disponible. Es algo que no se puede hacer en ningún Unix habitual y definitivamente nada que se pueda hacer con ningún sistema operativo propietario, ya que las empresas no están interesadas en facilitar sus códigos fuentes.

Linux cumple las especificaciones de Posix al 100%[2] y, debido a que el protocolo TCP/IP fue desarrollado alrededor de Unix, es ejecutado de manera nativa, la integración de los protocolos de Internet es perfecta; cuenta además con la ventaja de ser un sistema operativo basado en el proyecto GNU[3].

2.2. Breve historia.

2.2.1. ¿Qué es Linux?

Linux por naturaleza propia se puede decir que es el resultado de la frustración de los usuarios de minix, es único, no es un producto respaldado por una empresa de software, sino por miles de entusiastas usuarios alrededor del mundo, fue creado a principios de la década de los 90 utilizando los recursos de Internet para comunicarse entre sí y desarrollarlo. Linux es el nombre solo del Kernel, aunque comúnmente se utiliza para referirse al empaquetado, sistema completo o distribución, que son todos los programas que hacen posible la interacción con el usuario.

2.2.2. Linux y la GNU/GPL

Los sistemas operativos comerciales tienen un dueño, SUN es dueño de los derechos de Solaris, Apple de los de MacOS y Microsoft es dueño de los de Windows 2000, así que, ¿quién es dueño de los derechos de Linux?

Linux es un software del dominio público, muchos de sus componentes tienen derechos de autor. La mayoría de las utilidades están bajo GNU, de hecho, muchos de los que han contribuido con Linux han puesto su trabajo bajo la licencia GNU, de la cual se hablará más adelante.

Acerca del nacimiento de Linux se han escrito muchas cosas, la historia resulta apasionante, pero para poder entender la historia de Linux se debe conocer primero la historia de Unix, al cual se encuentra muy ligado.

---

[2] El nombre UNIX es una marca comercial, y para que un sistema sea nombrado UNIX tiene que pagar por el nombre. Sin embargo, la definición técnica de UNIX está concentrada en el estándar POSIX. Linux cumple al 100% con estos estándares, por tanto, si bien no podemos llamarlo UNIX, sí podemos llamarlo un sistema POSIX 100%, o un sistema similar a UNIX.

[3] Se explica más a detalle en la sección de licencias.

2.2.3. Unix

Las computadoras antes de los años 50, eran monousuarios, es decir, solo una persona podía trabajar en ellas a la vez, en el inicio de la década de los 60 aparecieron los sistemas de tiempo compartido, gracias a ello, varios usuarios podían hacer uso de la computadora al mismo tiempo. A mediados de los 60' el MIT y las empresas AT&T y General Electric se juntaron para realizar un gran proyecto, se trataba de hacer un Sistema Operativo de gran potencia al que denominaron MULTICS. El proyecto fue un fracaso pero uno de los programadores del MIT que había trabajado en el proyecto, Ken Thompson, y un grupo de colaboradores decidieron escribir una versión miniatura de MULTICS. Uno de los compañeros de Ken, Brian Kernigham, en una reunión de equipo bromeando llamó al sistema de Ken Thompson UNICS.- UNICS fue un gran éxito y Ken decidió que UNIX era un nombre más atractivo que UNICS. Había nacido UNIX.

Un famoso artículo[4] publicado en "*The Communications of the ACM"* en 1974 que describía UNIX atrajo la atención de las universidades que solicitaron el código fuente para estudiarlo y explicarlo en las aulas. Muy pronto, UNIX logró una gran aceptación en la comunidad científica y el interés por este sistema operativo comenzó a extenderse. A partir de este momento comienza una verdadera avalancha de versiones del sistema, lo que primero en un principio empezó como un proyecto de investigación se convirtió más tarde en un gran negocio.

Las más importantes de todas las versiones de UNIX fueron la BSD, de la Universidad de California en Berkeley, que contenía una serie de mejoras que hicieron a UNIX un sistema operativo más amigable, y la System V. Esta última surgió de la fusión de las respectivas versiones de UNIX de AT&T Bell Laboratories, los creadores del sistema, y Sun Microsystems. Actualmente el System V es considerado el estándar de UNIX, ya que toda la industria ha sido agrupada en torno a él.

2.2.4. Minix.

A pesar del éxito comercial de UNIX y de su aceptación como sistema operativo, el código fuente de UNIX no podía ser explicado en aulas universitarias, de modo que el desarrollo de sistemas operativos volvía a ser una ciencia restringida a un reducido grupo de empresas y personas. Ante esta situación, el profesor Andrew Tanembaum, de la Universidad de Vrije, en Amsterdam, decidió imitar a Ken Thompson cuando escribió el código de UNIX basándose en MULTICS, e inspirándose en UNIX llevó a cabo un nuevo sistema operativo mucho más reducido, al que llamó MINIX (de Mini-UNIX). MINIX había sido desarrollado en una IBM PC y, sin embargo ofrecía las mismas llamadas al sistema que UNIX V7.

---

[4] The UNIX Time-Sharing System. D. M. Ritchie and K. Thompson

2.2.5. Linux.

En 1990, Linus Torvalds, un estudiante de 23 años de la Universidad de Helsinki, en Finlandia, comenzó a desarrollar, como hobby, un proyecto basado en el MINIX de Andrew Tenembaum. Quería llevar a cabo, sobre una computadora con procesador Intel 80386, un sistema operativo tipo UNIX que ofreciera más capacidades que el limitado MINIX, que sólo se usaba para enseñar una cierta filosofía de diseño. Quería aprovechar la arquitectura de 32 bits, las propiedades de conmutación de tareas que incorporaba la interfaz en modo protegido del 80386 y eliminar las barreras del direccionamiento de memoria.

Linus empezó escribiendo el núcleo del proyecto en ensamblador, y luego comenzó a añadir código en C, lo cual incrementó la velocidad de desarrollo, e hizo que empezara a tomarse en serio su idea de hacer un "MINIX mejor que MINIX". La primera versión, la 0.01 no tenía driver de disquete, y ni siquiera la dio a conocer. Llevaba incorporado un pequeño sistema de archivos y un driver de disco con mucho errores... pero funcionaba. En octubre de 1991, anunció la primera versión "oficial" de LINUX, la 0.02, que ya era capaz de ejecutar el bash y el compilador gcc de GNU.

En comp.os.minix, un foro de discusión en Internet acerca del sistema operativo de Tenembaum, Linus Torvalds escribió un llamamiento que comenzaba con una famosa frase (según algunos):

> *¿Añoras los maravillosos días del MINIX-1.1, cuando los hombres eran hombres y escribían sus propios drivers? ¿Careces de proyectos interesantes y te mueres por desafiar a un sistema operativo que puedas modificar a tu antojo? ¿Te resulta frustrante que todo funciones con MINIX? ¿Estás harto de trasnochar para poder conseguir que funcione un programa?*
> *Entonces, esta carta puede ser justamente para ti.*
> *Como comenté hace un mes, estoy trabajando en una versión libre de un sistema tipo MINIX para computadoras AT-386. Finalmente ha sido mejorado el entorno, que incluso se puede utilizar, y estoy deseoso de sacar las fuentes de una distribución más potente.*
> *Es solo la versión 0.02... pero ha conseguido que funcione bien bash, gcc, gnu-make, gnu-se, compress, etc., bajo él"* [5]

Según otros, lo primero que Linus envió fue lo siguiente

Extracto del grupo de noticias comp.os.minix de Usenet.

[5] Historia de Linux – Linux Users Group Tucuman, en esta página no se muestra la fuente del mensaje, ni los fuentes de los cuales se obtuvo la traducción, pero es uno de los primeros comentarios que escuché acerca del nacimiento de Linux.

Fecha: 3 Jul 91 10:00:50 GMT
Mensaje enviado por Linus Torvalds

Hola Ciudadanos de la red,

Debido a un proyecto en el que estoy trabajando (en minix), estoy interesado en la definición del estándar POSIX. ¿Habrá alguien que me pueda indicar en dónde encuentro (de preferencia) en formato legible por una computadora las reglas de POSIX? un sitio de FTP sería muy bueno.[6]

A partir de ese momento muchos "netlanders"[7] programadores, beta testers[8], etc., apoyaron el proyecto de Linus, así es como se inició lo que ahora conocemos como Linux. Pero eso era sólo el principio, es decir, había planteado lo que es el Kernel, falta lo que son los programas que se comunican con el usuario, es decir, faltaban los front-end, es ahí donde entran los programas que se amparan bajo la licencia GNU, pero ese tema se tratará más adelante.

Después de la versión 0.03, Linus saltó en la numeración hasta la 0.10, más y más programadores a lo largo y ancho de Internet empezaron a trabajar en el proyecto y después de sucesivas revisiones, Linus incrementó el número de versión hasta la 0.95 (Marzo 1992). Mas de un año después (Diciembre 1993) el núcleo del sistema estaba en la versión 0.99. Linus Torvalds continuó con el desarrollo hasta que el 14 de Marzo de 1994 liberó el Kernel de Linux versión 1.0, la primera versión considerada estable.

El proyecto no está completo: Miles de programadores diseminados por todo el mundo, amplían y mejoran Linux continuamente y lo portan a otras plataformas (Alan Cox es actualmente una persona clave en el desarrollo del Kernel). La versión actual más estable es la 2.4.5 de mayo del 2001 y por supuesto los desarrollos públicos continúan a cargo de numerosas organizaciones y programadores independientes. Linus Torvalds, además de seguir programando, se ha dedicado desde entonces a recopilar, aceptar, desechar, normar y organizar código de programación con el que le contribuyen otros programadores y a orientar y unir en equipos a grupos de programadores con ideas afines.

Linux es el sistema operativo que más inquietud ha despertado en los últimos años en el mundo de la Computación. Lo que empezó en 1991 como un hobby para un joven estudiante finlandés llamado Linus Torvalds, ha pasado de ser un(a) juguete(pasión) de hackers a una herramienta importante tanto para usuarios particulares como profesionales. Hace sólo unos años (1992), había unos 100,000 usuarios de Linux aproximadamente. Hoy, aproximadamente 16 millones de usuarios dependen de Linux (Noviembre 2000, Fuente: Linux Counter) para

[6] Historia de Linux, Pagina de Linux PPP
[7] Ciudadanos de la red.
[8] Probadores de programas de los que aun no se libera la versión final

administrar finanzas, usar y controlar servicios de Internet, diseño gráfico, desarrollo de aplicaciones científicas y mucho más. Este número crece rápidamente. Cada día, nuevos usuarios descubren las potencialidades de Linux.

2.3. Licencias

Richard Stallman es el fundador de la licencia GNU[9], Linus Torvalds es el creador de un Kernel 'Unix – Like', la pregunta es, como se relacionan estos dos personajes? La respuesta se encuentra en el aprovechamiento de las ventajas que brinda la licencia GNU, como a continuación se explica.

2.3.1. Orígenes

El origen de la licencia GNU se remonta a lo que era el antecesor del software libre[10], que era cuando no estaba limitado el compartir el software, era algo tan antiguo como las computadoras mismas, de la misma manera en que compartir recetas era tan antiguo como cocinar. No denominábamos «software libre» a nuestro software porque dicho término no existía; pero eso es lo que era. Cuando alguien de otra universidad o compañía deseaba portar y usar un programa, lo permitíamos con gusto. Si usted veía a alguien usando un programa interesante y poco conocido, siempre se podía pedir el código fuente para verlo, de manera que uno podía leerlo, cambiarlo, o canibalizar ciertas partes del mismo para hacer un nuevo programa.[11]

El software libre tiene su origen en las grandes universidades, Richard Stallman era un investigador de Laboratorio de Inteligencia Artificial del MIT y a partir de que las grandes empresas comenzaron a vender sus máquinas con sus propios sistemas operativos y les hacían firmar un <<acuerdo de no revelar>> ((*nondisclosure agreement)*. De acuerdo al espíritu prevaleciente en esa época eso significaba no poder ayudar a los vecinos. El proyecto GNU nace de la inquietud de hacer un sistema operativo libre, se podía tener de nuevo una comunidad de hackers cooperando. Decidió que el sistema operativo fuera compatible con Unix pues así sería portable, además de que los usuarios de Unix podrían cambiarse a el con facilidad.

2.3.2. Copyleft, GNU/GPL

Esta Licencia se suele denominar GNU Copyleft, un juego de palabras con la palabra copyright. Cubre todo el software producido por el proyecto GNU de la Free Software Foundation. La licencia permite a los programadores crear software para todos. La premisa básica de GNU es que el software deberá estar disponible para todos y que sí alguien desea modificar el programa para sus propios fines,

---

[9] . El nombre GNU se eligió siguiendo una tradición hacker, como acrónimo recursivo para «*GNU's Not Unix*». El Proyecto GNU - Fundación para el Software Libre (Free Software Foundation (FSF)

[10] Free Software – Traducción literal.

[11] ídem 9

esto debe ser posible. La única condición es que no puede limitarse el código modificado, los demás tienen derecho a utilizar el nuevo código. La GNU Copyleft, o GPL, permite a los creadores de un programa conservar sus derechos de autor legales, pero permite a los demás copiar, modificar y vender el nuevo programa resultante. Sin embargo, al hacerlo no pueden limitar ningún derecho similar a los que compren el software. Si vende el programa tal como está, o una modificación del mismo, debe facilitar el código fuente. Por este motivo Linux viene con el código fuente completo.

Así, el proyecto GNU se centró en hacer todas las aplicaciones para un sistema operativo que tendría por Kernel a HURD basado en un microkernel llamado mach, el cual es un micronúcleo desarrollado en la Universidad de Carnegie Mellon y luego en la universidad de Utah, HURD, es una colección de servidores que corren sobre mach y se hacen cargo de las tareas del Kernel.

Pero el HURD no está listo, es en este punto cuando entra el Kernel de Linux, compatible con Unix, así, se combino lo que es el Kernel que al proyecto GNU le hacia falta con el sistema GNU, nace lo que oficialmente se llama GNU/Linux[12] o simplemente Linux.

[12] Se denomina a esta versión GNU/Linux para expresar su composición como combinación de un sistema GNU con Linux como núcleo

Capítulo 3. IPv6

Introducción.

A principios del siglo XXI, Internet es una red dramáticamente diferente de lo que era en los años 80´s y se ha convertido en la red pública de datos más grande. Así, la necesidad de redes privadas está creciendo rápidamente, hay muchos factores que motivan este incremento, en general se reconocen tres ramas principales, esas son, el continuo crecimiento en el uso de PC's para el hogar, la rápida introducción de dispositivos inteligentes y él (fenomenal) crecimiento de negocios en la red y las telecomunicaciones.

Uno de los motivos básicos por el que surge en el seno del IEFT la necesidad de crear un nuevo protocolo, que en un principio se denomino IPng o Siguiente Generación del Protocolo de Internet, fue la evidente falta de direcciones. IPv4 tiene un espacio de direcciones de 32 bits, es decir 2^{32}; en cambio, IPv6 ofrece un espacio de direcciones de 2^{128}.

Pero, ¿por qué el espacio de direcciones se está terminando?, ¿Se debió acaso a un error en el diseño de sus creadores?, en realidad lo que sucedió fue que las tecnologías de información han evolucionado de un modo mucho más explosivo de lo esperado, es decir, nunca se imaginaron que sus protocolos y descubrimientos se iban a aplicar en multitud de campos, no sólo científicos y de educación, sino también en innumerables facetas de la vida cotidiana.

Lo anterior conlleva otras aplicaciones para las que originalmente fue creado, así que ha sido necesario colocar parches al protocolo básico, entre los más importantes se encuentran la calidad de servicio (QoS[1]), seguridad (IPSec[2]) y movilidad, fundamentalmente.

La ventaja mas evidente de IPv6 es el espacio de direcciones, como se puede apreciar a primera vista. Algunas de las soluciones planteadas al problema del espacio de direcciones, el cual se originó en la década de los 80's a causa de

[1] Quality of service, Calidad de servicio.
[2] IPSec es un grupo de extensiones de la familia del protocolo IP. IPSec provee servicios criptográficos de seguridad. Estos servicios permiten la autenticación, integridad, control de acceso, y confidencialidad

la delegación de direcciones sin ningún tipo de optimización dejando grandes espacios discontinuos; seria la renumeración y reasignación de dicho espacio de direcciones. Sin embargo es una tarea impensable, requiere de esfuerzos de coordinación a escala mundial. Además, el problema de las inmensas tablas de ruteo en las troncales de Internet no se resuelve, lo hace ineficaz y perjudica enormemente los tiempos de respuesta.

3.1. El problema de las IP's validas.

El problema no es igual en todos los puntos del planeta, es decir, en Norteamérica se puede decir que es inapreciable, sin embargo, en zonas geográficas como Asia el problema está llegando a ser grave, por ejemplo, China solicitó direcciones IP para conectar 60,000 escuelas y sólo consiguió una clase B (65,535 direcciones), países europeos y africanos que solo tienen una clase C (255 direcciones) para todo el país. En países como Japón o en Europa el problema es creciente, dado el importante desarrollo en telefonía celular, comunicación inalámbrica, módems, etc., que requieren una IP homologada para aprovechar al máximo sus posibilidades e incrementar el número de aplicaciones para las que pueden ser empleados.

De igual manera, la utilización de direcciones está en un rango de 1:10 (una dirección por 10 usuarios) y tiende a ser 1:1 en el presente y se presume que en el futuro cada usuario puede llegar a tener hasta 50 o 100 (50:1 ó 100:1).

En 1997, el mercado de dispositivos con aplicaciones capaces de conectarse a Internet (sin incluir terminales ni computadoras, solamente WebTV, agendas electrónicas, teléfonos celulares, PDA's, etc.), era de 3,000,000. En 1998, este se duplica hasta llegar a los 6,000,000, y las previsiones de crecimiento para este año según IDC[3] son de 56,000,000. A lo anterior, si le agregamos el crecimiento de la nueva generación de telefonía móvil (UMTS)[4], en el año 2003 se prevén cifras del orden de los 1.000.000.000, lo mismo que para la telefonía fija y que para los usuarios fijos de Internet, en ese momento, los usuarios móviles de Internet se acercarán a los 400.000.000.

En el 2005, para conectar dispositivos de red, no dispositivos de usuario, se prevé se necesitarán 3.2 millones y de 6.3 para el 2010, aunado a esto si le sumamos los nuevos dispositivos o los ya existentes a los que se les dan nuevas o mejoradas aplicaciones mediante la conexión a la red.

[3] Revista especializada en análisis de los mercados de Internet
[4] UMTS, siglas que en inglés hacen referencia a los Servicios Universales de Telecomunicaciones Móviles

Tecnologías emergentes como redes inalámbricas, Bluetooth[5], WAP (Wireless Aplication Protocol), etc., que hace patente el crecimiento, al menos, en lo que número de direcciones se refiere.

Por ejemplo, la última tendencia es la de permitir a cualquier dispositivo serie ser conectado a una LAN o una WAN, y también a Internet. Los dispositivos de este tipo, llamados “Universal Device Server”, permiten que aplicaciones impensables por las limitaciones de los cableados serie se conecten y transmitan información con lujo de detalles.

El uso de mecanismos de NAT, de seguir con IPv4, sólo retrasa lo inevitable, de ser una solución temporal pasaría a ser “invariable permanentemente”, no obstante que ello implica la imposibilidad práctica de muchas aplicaciones que quedan relegadas al uso de intranets, ya que muchos protocolos no son capaces de atravesar los dispositivos NAT.

3.2. Historia de IPv6.

Este es un capítulo muy amplio en lo referente a las discusiones y revisiones de las propuestas hechas por el IEFT y el IAB, por lo que sólo se especifican a grandes rasgos los hechos que dieron nacimiento a lPv6.

Hay tres propuestas principales para IPng[6]:

- **CATNIP (Common Architecture for the Internet)** es el desarrollo de un viejo protocolo (TCP/IX) que integra IPv4, Novell IPX y OSI CLNP(Connectionless Networking Protocol) y proporciona una infraestructura común. Se acerca en diseño a CLNP y enfatiza la facilidad de interoperabilidad con las implementaciones existentes de los tres. El paquete CATNIP contiene toda la información requerida por los tres protocolos en un formato comprimido usando una cabecera de 16 o más bytes. CATNIP usa una dirección de longitud variable. Las direcciones IPv4 existentes se mapean a direcciones de 7 bytes de las cuales sólo los últimos 4 bytes son la dirección IPv4. Los host IPv4 existentes deberían limitarse a ínteroperar de esta forma con los hosts CATNIP. CATNIP se describe en el [RFC1707].

- **TUBA(TCP and UDP with Bigger Addresses)** también se basa en CLNP; en pocas palabras, sustituye a IPv4 en la pila TCP/IP. Enfatiza las redes multiprotocolo. La transición entre IPv4 a IPng se hace usando una estrategia de pila dual. La pila de protocolo tiene dos capas de red independientes y cuando se intenta comunicar con otro host, un host de pila

[5] La tecnología Bluetooth es una especificación abierta para la comunicación inalámbrica (WIRELESS) de datos y voz. Está basada en un enlace de radio de bajo costo y corto alcance, implementado en un circuito integrado de 9 x 9 mm, proporcionando conexiones instantáneas (ad hoc) para entornos de comunicaciones tanto móviles como estáticos

[6] Tutorial y descripción técnica -- ipng ulpgc historia de ipv6

dual consulta al DNS sobre la dirección IP y el NSAP (Network Service Access Point) que es el equivalente CLNP. Si el DNS devuelve tanto la dirección IP como el NSAP, el host se comunica usando CLNP como protocolo de red. TUBA se describe en el [RFC1347]. Ver también el [RFC1526], [RFC1561].

- **SIPP(Simple Internet Protocol Plus)** es una combinación del esfuerzo de tres grupos de trabajo anteriores del IETF dedicados al desarrollo de IPng.

    - **IPAE (IP Address Encapsulation)** determina que las extensiones de IPv4 lleven direcciones más largas, y cómo debe realizarse la transición de una versión a otra.
    - **SIP (Simple Internet Protocol)** es una sustitución de IPv4 con una cabecera IP simplificada y direccionamiento de 64 bits, unido a IPAE usando la cabecera SIP y los mecanismos de transición IPAE.
    - **PIP ("P" Internet Protocol)** fue una nueva marca para un protocolo de Internet, diseñado con una amplia variedad de características avanzadas y usando direccionamiento de longitud variable. PIP se unió con SIP cuando quedó patente que las mejores características de PIP se podían usar con el esquema de direccionamiento de 64 bits de SIP y los mecanismos de transición de IPAE.

  SIPP es un desarrollo evolutivo de IPv4. Enfatiza la eficiencia de operatividad en una gran variedad de tipos de red y la facilidad de interoperabilidad. Además del direccionamiento de 64 bits, incluye el concepto de direcciones extendidas usando una opción de encaminamiento: la longitud de la dirección efectiva puede ser cualquier múltiplo de 64. SIPP se describe en el [RFC1710].

3.3. Principales características de IPv6.

El resumen de las principales características de IPv6 se muestra a continuación:

- Un mayor espacio de direcciones
- “Plug and Play”: Es auto configurable
- La seguridad intrínseca en el núcleo del protocolo que le proporciona IPsec
- Calidad de servicio (QoS) y Clase de Servicio (CoS)
- Multicast: Envío de un mismo paquete a un grupo de receptores
- Anycast: Envío de un paquete a un receptor dentro de un grupo.
- Paquetes IP eficientes y extensibles, sin fragmentación de los routers, alineados a 64 bits para el óptimo procesamiento con los nuevos procesadores de 64 bits, y un encabezado de longitud fija.
- Paquetes de datos con posibilidad de carga útil de hasta 65,535 bytes.
- El enrutado se hace mas eficiente debido a la jerarquía de direccionamiento basada en la agregación.

- Renumeración y “multihoming”[7], que facilita el cambio de proveedor de servicios.
- Movilidad.

Hay que resaltar que estas son solamente algunas características básicas, la estructura propia del protocolo permite que éste sea escalado según las necesidades y los requerimientos de las aplicaciones, y además su principal carta frente a IPv4 su escalabilidad.

3.4. Especificaciones básicas de IPv6

Para visualizar los cambios que se realizan en la estructura del datagrama se presenta el encabezado de IPv4 con dos tonalidades diferentes de gris, el gris mas oscuro desaparece, el otro se mantiene como se muestra a continuación:

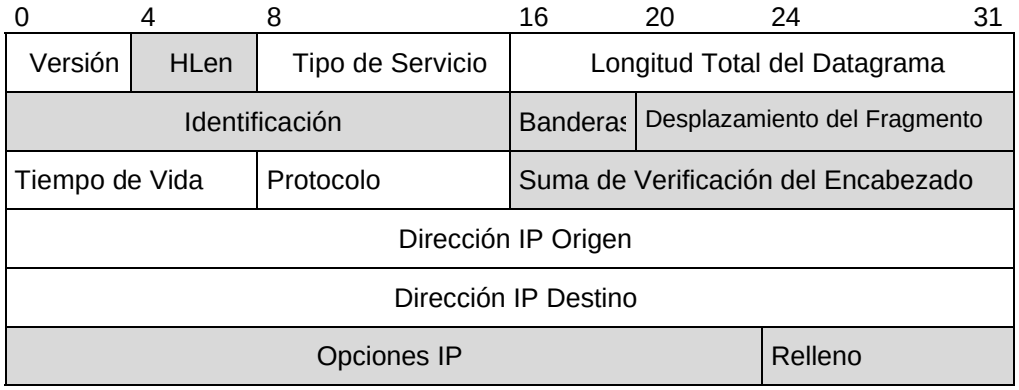

Figura 1. Datagrama IPv4 que muestra que campos se modifican y cuales desaparecen

La descripción de los campos se presenta en el capitulo 1, por lo que no se reproduce aquí.

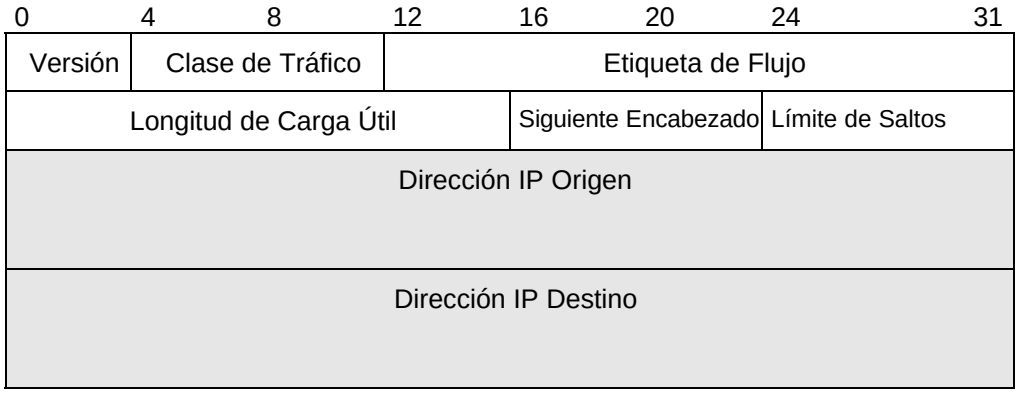

Figura 2. Datagrama de IPv6

[7] Capacidad de un servidor Web para tener más de una dirección de Internet y más de una página principal

De acuerdo a la figura 1, el datagrama IP pasa de 12 campos fijos y dos opcionales en el datagrama original a solo 8 (figura 2). La razón principal de que esto suceda es la redundancia en la que se cae al proporcionar la misma información en diferentes formas, por ejemplo, la suma de verificación, que es realizado por otros mecanismos de encapsulado (IEEE 802 MAC, framing PPP, capa de adaptación ATM, etc.).

La descripción de la funcionalidad y significado de los campos se muestran a continuación:

- Versión: Es un número igual a 6 es la versión del protocolo de Internet, ocupa 4 bits

- Clase de Tráfico: Campo de la clase de tráfico, también denominado como prioridad (priority) o simplemente clase (class), se puede decir que es el equivalente de TOS en IPv4, tiene una longitud de 8 bits (un byte).

- Etiqueta de Flujo: Se denomina Flow Label, se ocupa para permitir tráfico con requisitos en tiempo real. Tiene una longitud de 20 bits.

- Longitud de la carga Útil: Entero sin signo de 16 bits (2 bytes) se llama también como Payload Length, se refiere a la longitud de los datos y puede ser de hasta 65536 bytes (2^{16}).

- Siguiente Encabezado: Se conoce también como Next Header, en lugar de usar encabezados de longitud variable se utilizan encabezados sucesivos encadenados, tiene una longitud de 8 bits (1 byte).

- Límite de Saltos: Se conoce también como Hop Limit, entero sin signo de 8 bits (1 byte) que se decrementa en 1 por cada nodo que reenvía el paquete, si el Límite de Saltos llega a cero el paquete se descarta.

- Dirección IP Origen: Dirección de 128 bits del origen del paquete.

- Dirección IP Destino: Dirección de 128 bits del recipiente pretendido del paquete (posiblemente no el último recipiente, si está presente un encabezado de Enrutamiento).

Los campos de Calidad de Servicio (QoS) y Clase de Servicio (CoS) funcionan como un poderoso mecanismo de control de flujo y de asignación de prioridades diferenciadas según los tipos de servicio.

La longitud del encabezado se duplica, es decir, es de 40 bytes, la ventaja es que se eliminan los campos duplicados, además la longitud fija del encabezado implica una mayor capacidad de procesamiento de los ruteadores por medio del

hardware, lo que significa una mayor eficiencia, también la alineación de los campos a 64 bits, da como resultado que los nuevos procesadores y microcontroladores puedan procesar eficazmente el encabezado.

3.5. Encabezados de extensión de IPv6

En IPv6, la información opcional de la capa de Internet es codificada en encabezados separados que pueden ser colocados entre el encabezado de IPv6 y el encabezado de la capa superior en un paquete. Hay un pequeño número de dichos encabezados de extensión, cada uno identificado por un valor de Encabezado Siguiente distinto. Como se ilustra en la figura 4, un paquete de IPv6 puede acarrear cero, uno o más encabezados de extensión, cada uno identificado por el campo de próximo encabezado del encabezado precedente.

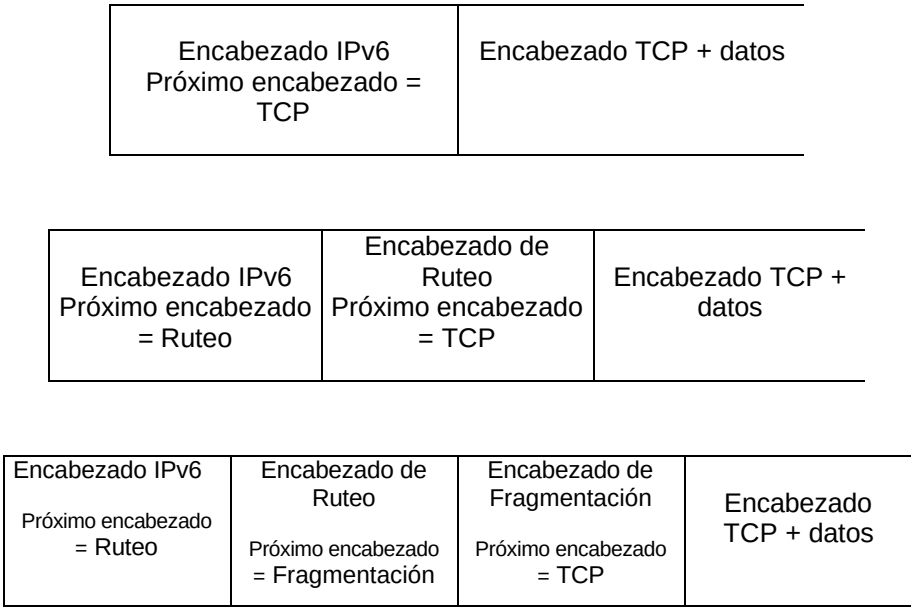

Figura 3. Ejemplo de encabezados de extensión

Con una excepción, los encabezados de extensión no son examinados o procesados por ningún nodo a lo largo de la ruta de entrega del paquete, mientras el paquete alcance el nodo (o cada uno del conjunto de nodos, en el caso de multicast) identificado en el campo de Dirección IP Destino del encabezado de IPv6. Ahí, la demultiplexación normal en el campo de Próximo encabezado del encabezado IPv6 invoca al módulo a procesar del primer encabezado de extensión, o el encabezado de nivel superior si no hay encabezados de extensión presentes. Los contenidos y semánticas de cada encabezado de extensión determinan si se procede o no al próximo encabezado. Consecuentemente, los encabezados de extensión deben ser procesados estrictamente en el orden en que aparecen en el paquete; un receptor no debe, por ejemplo, explorar a través

de un paquete buscando un particular tipo de encabezado de extensión y procesar dicho encabezado antes que procesar todas las precedentes.

La excepción referida en el párrafo precedente es el encabezado de opciones Salto a Salto, la cual acarrea información que debe ser examinada y procesada por todos los nodos a lo largo de la ruta de entrega del paquete, incluyendo los nodos origen y destino. El encabezado de opciones Salto a Salto, cuando está presente, debe seguir inmediatamente al encabezado de IPv6. Su presencia es indicada por el valor cero en el campo de próximo encabezado en el encabezado IPv6.

Si como resultado de procesar un encabezado, un nodo requiere proceder al próximo encabezado pero el valor del próximo encabezado en el encabezado actual no es reconocido por el nodo, debe descartar el paquete y enviar un mensaje ICMP de Problema de Parámetro al origen del paquete, con un código de valor ICMP de 1 (“Encontrado un tipo de encabezado irreconocible”) y el puntero ICMP conteniendo el paquete original. La misma acción debe ser tomada si un nodo encuentra un valor de próximo encabezado igual a cero en cualquier otro encabezado de IPv6.

Cada encabezado de extensión es un múltiplo entero de 8 bytes de largo, para conservar una alineación de 8 bytes para encabezados subsecuentes. Los campos multibyte dentro de cada encabezado de extensión son alineados en sus límites naturales, por ejemplo, los campos de n bytes de ancho son colocados en un múltiplo entero de n bytes desde el inicio del encabezado, para n = 1, 2, 3, 4 ú 8.

Una implementación completa de IPv6 incluye los siguientes encabezados de extensión:

- Opciones Salto a Salto (Hop-by-Hop Options)
- Ruteo (Tipo 0) (Routing)
- Fragmentación (Fragment)
- Opciones de Destino (Destination Options)
- Autenticación (Authentication)
- Seguridad del Encapsulado de la Carga Útil (Encapsulating Security Payload)

3.5.1. Orden de los encabezados de extensión.

Cuando más de un encabezado de extensión es usado en el mismo paquete, es recomendado que esos encabezados aparezcan en el orden siguiente:

1. Encabezado IPv6
2. Encabezado de Opciones Salto a Salto

3. Encabezado de Opciones de Destino
4. Encabezado de Ruteo
5. Encabezado de Fragmentación
6. Encabezado de Autenticación
7. Encabezado de Seguridad del Encapsulado de la Carga útil
8. Encabezado de Opciones de Destino
9. Encabezado de Capa de Nivel Superior

Cada encabezado de extensión debe ocurrir al menos una vez, excepto para el encabezado de Opciones de Destino el cual debe ocurrir al menos un par de veces (una vez antes del encabezado de Ruteo y una antes del encabezado de Capa de Nivel Superior).

Si el encabezado de Capa de Nivel Superior es otro encabezado IPv6 (en el caso de que IPv4 este siendo tuneleado o encapsulado en IPv6), puede ser seguido por su propio encabezado de extensión, los cuales separadamente están sujetos al mismo orden de recomendaciones.

Siempre y cuando otros encabezados de extensión sean definidos, las restricciones de orden relativas concernientes a los encabezados listados arriba deben ser especificados.

Los nodos deben aceptar e intentar procesar encabezados de extensión en cualquier orden y ocurriendo cualquier número de veces en el mismo paquete, excepto por el encabezado de Opciones Salto a Salto el cual es restringido para aparecer inmediatamente después de solamente un encabezado IPv6. No obstante, es altamente recomendado que las fuentes de paquetes IPv6 se adhieran al orden de recomendaciones antes mencionado, mientras y a menos que, en especificaciones subsecuentes, sea revisada dicha recomendación.

3.5.2. Opciones

Dos de las actuales definiciones de los encabezados de extensión – el encabezado de Opciones Salto a Salto y el encabezado de Opciones de Destino – acarrean un número variable de opciones codificadas tipo-longitud-valor (TVL) de la siguiente forma:

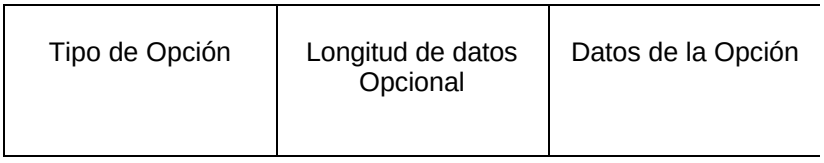

Figura 4. Muestra de TVL

Donde:

Tipo de Opción	Identificador de 8 bits del tipo de opción
Longitud de datos Opcional	Entero sin signo de 8 bits. Longitud del campo de Datos de la Opción, en bytes
Datos de la Opción	Campo de longitud variable. Datos Específicos del tipo de Opción

La secuencia de las opciones dentro de un encabezado debe ser procesada estrictamente en el orden en que aparecen en el encabezado; un receptor no debe, por ejemplo, buscar a través del encabezado buscando un particular tipo de opción y procesar dicha opción antes que procesar todas las precedentes.

Los identificadores Tipo de Opción son codificados internamente tal que sus dos bits de más alto orden especifiquen la acción que debe ser tomada si el nodo que procesa IPv6 no reconoce el Tipo de Opción:

00 – Omite esta opción y continua procesando el encabezado

01 – Descarta el paquete

10 – Descarta el paquete e, independientemente de que la Dirección IP Destino sea o no multicast, envía un mensaje ICMP de Problema de Parámetro, código 2, a la Dirección IP Origen del paquete, señalando el Tipo de Opción irreconocible

11 – Descarta el paquete y, sólo si la Dirección IP Destino del paquete no era una dirección multicast, envía una mensaje ICMP de Problema de Parámetro, código 2, a la Dirección IP Origen del paquete, señalando el Tipo de Opción irreconocible

El tercer bit de más alto orden del Tipo de Opción especifica si los Datos de Opción pueden o no cambiar el enrutado al destino final del paquete. Cuando un encabezado de Autenticación está presente en el paquete, para cualquier opción cuyos datos puedan cambiar el enrutado, esta opción entera debe ser tratada como bits de valor cero cuando se computen o se verifique el valor de autenticación del paquete.

0 – Los Datos de la Opción no cambian el enrutado
1 – Los Datos de la Opción pueden cambiar el enrutado

Los tres bits de más alto orden descritos anteriormente deben ser tratados como una parte del Tipo de Opción, no independientemente del Tipo de Opción. Esto es, una opción particular es identificada por un Tipo de Opción completo de 8 bits, no solamente por los 5 bits de más bajo orden de un Tipo de Opción.

El mismo espacio de numeración de Tipo de Opción se usa tanto para el encabezado de Opciones Salto a Salto como para el encabezado de Opciones de

Destino. Sin embargo, la especificación de una opción particular puede restringir su uso a solamente uno de esos dos encabezados.

Las opciones individuales pueden especificar requerimientos de alineación, para asegurar que los valores multi – bytes dentro de los Datos de la Opción caen dentro de los límites naturales. El requerimiento de alineación de una opción es especificado usando la notación $xn + y$, lo que significa que el Tipo de Opción debe aparecer como un múltiplo entero de x bytes del inicio del encabezado, más y bytes. Por ejemplo:

$2n$	Significa cualquier desplazamiento de 2 bytes a partir del inicio del encabezado
$8n + 2$	Significa cualquier desplazamiento de 8 bytes a partir del inicio del encabezado, más dos bytes

Hay dos opciones de relleno las cuales son usadas cuando es necesario alinear opciones subsecuentes y rellenar el encabezado contenedor con un múltiplo de 8 bytes de longitud. Esas opciones de relleno deben ser reconocidas por todas las implementaciones de IPv6.

Opción Pad1 (requerimiento de alineación: ninguno)

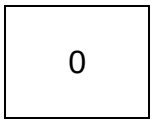

Figura 5. Ejemplo de Pad1

Nota: el formato de la opción Pad1 es un caso especial – no tiene los campos de longitud y de valores.

La opción Pad1 es usada para insertar un byte de relleno dentro del Área de Opciones de un encabezado. Si más de un byte de relleno es requerido, la opción PadN, descrita a continuación, debe ser usada, en vez de múltiples opciones Pad1.

Opción PadN (requerimiento de alineación: ninguno)

1	Longitud de Datos Opcional	Datos de la Opción

Figura 6. Ejemplo de PadN

La opción PadN es usada para insertar dos o mas bytes de relleno dentro del área de Opciones del encabezado. Para N bytes de relleno, el campo Longitud de

Datos Opcional contiene el valor $N-2$, y los Datos de la Opción consisten de $N-2$ bytes con valor cero.

3.5.3. Encabezado de Opciones Salto a Salto.

El encabezado de Opciones Salto a Salto es usado para acarrear información opcional que debe ser examinada por todos los nodos a lo largo de la ruta de entrega del paquete. El encabezado de Opciones Salto a Salto es identificado por un valor de Próximo Encabezado de 0 en el encabezado IPv6, y tiene el siguiente formato:

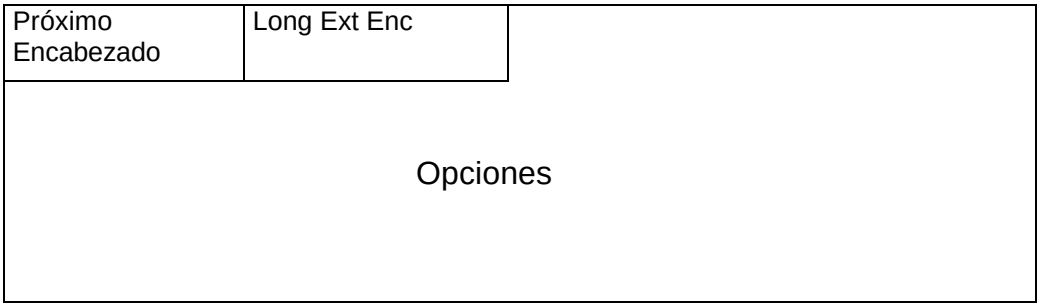

Figura 7. Encabezado de Opciones Salto a Salto

Próximo Encabezado	Selector de 8 Bits. Identifica el tipo de encabezado inmediatamente siguiente al encabezado de Opciones Salto a Salto. Usa los mismos valores del campo protocolo de IPv4 [RFC1700].
Long Ext Enc	Entero de 8 bits sin signo. Longitud del encabezado de Opciones Salto a Salto en unidades de 8 bytes, no incluye los primeros 8 bytes.
Opciones	Campo de longitud variable, de longitud tal que el encabezado de Opciones Salto a Salto es un múltiplo entero de 8 bytes de largo. Contiene una o más opciones TLV codificadas.

3.5.4. Encabezado de Ruteo

El encabezado de Ruteo es usado por un origen IPv6 para listar uno o más nodos intermedios a ser “visitados” en la ruta de destino de un paquete. Esta función es muy similar a las opciones de Fuente Imprecisa y Registro de Ruta de IPv4. El encabezado de Ruteo es identificado por un próximo encabezado con valor 43 en el encabezado inmediato precedente, y tiene el siguiente formato:

Próximo Encabezado	Long Ext Enc	Tipo de Ruteo	Segmentos Restantes
Datos específicos del tipo			

Figura 8. Encabezado de Ruteo

Próximo Encabezado	Selector de 8 Bits. Identifica el tipo de encabezado inmediatamente siguiente al encabezado de Ruteo. Usa los mismos valores del campo protocolo de IPv4 [RFC1700].
Long Ext Enc	Entero de 8 bits sin signo. Longitud del encabezado de Ruteo en unidades de 8 bytes, no incluye los primeros 8 bytes.
Tipo de Ruteo	Identificador de 8 bits de una variante en particular del encabezado de Ruteo
Segmentos Restantes	Entero de 8 bits sin signo. Número de segmentos de ruteo restantes, por ejemplo, el número explícitamente listado de nodos intermedios que todavía tienen que ser visitados antes de alcanzar el destino final.
Datos específicos del tipo	Campo de longitud variable, de formato determinado por el Tipo de Ruteo, y de longitud tal que el encabezado de Ruteo completo es un múltiplo entero de 8 bytes de largo

Si mientras se procesa un paquete recibido, un nodo encuentra un encabezado de Ruteo con un valor irreconocible de Tipo de Ruteo, el comportamiento requerido del nodo depende del valor del campo de Segmentos Restantes, como se muestra:

Si Segmentos Restantes es igual a cero, el nodo debe ignorar el encabezado de Ruteo y proceder a procesar el próximo encabezado en el paquete, cuyo tipo es identificado por el campo Próximo Encabezado en el Encabezado de Ruteo.

Si Segmentos Restantes no es cero, el nodo debe descartar el paquete y enviar un mensaje ICMP de Problema de Parámetro, Código 0, a la Dirección IP Origen del paquete, señalando el Tipo de Ruteo irreconocible.

Si, después de procesar un encabezado de Ruteo de un paquete recibido, un nodo intermedio determina que el paquete debe ser reenviado sobre un enlace cuyo MTU es menor al tamaño del paquete, el nodo debe descartar el paquete y enviar un mensaje ICMP de Paquete Muy Grande a la Dirección IP Origen del paquete.

El encabezado de Ruteo tipo 0 tiene el siguiente formato:

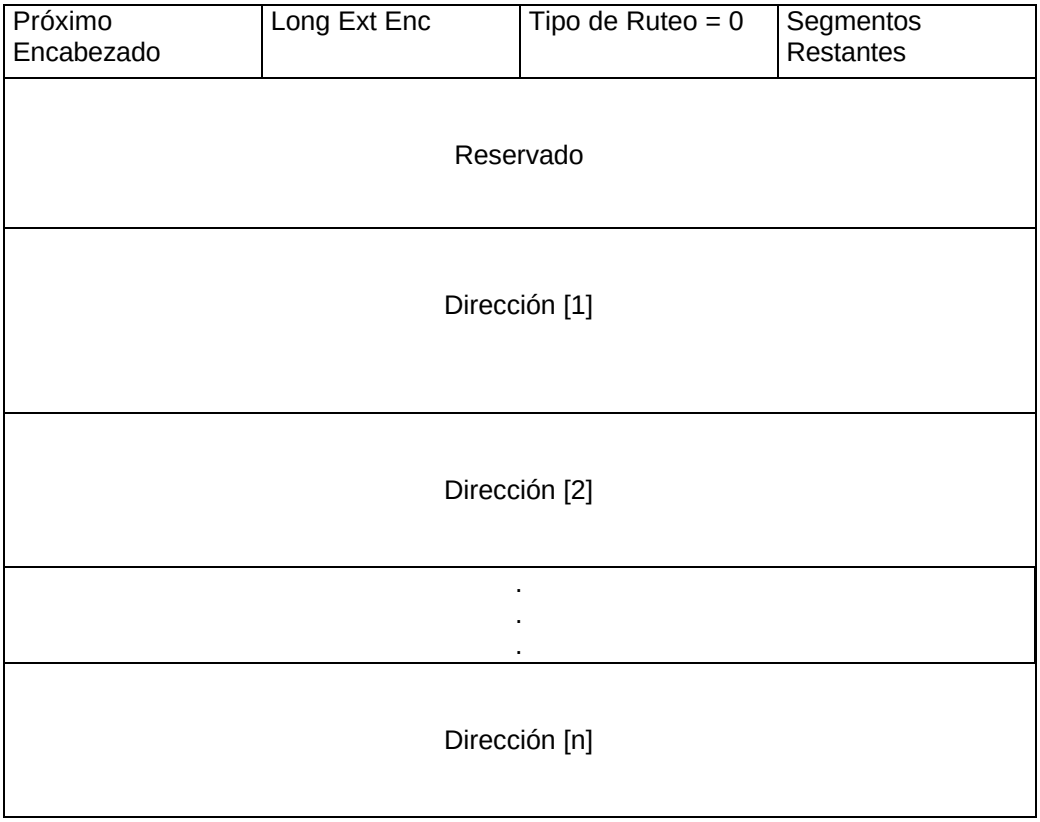

Figura 9. Ejemplo encabezado de Ruteo Tipo 0

Próximo Encabezado	Selector de 8 Bits. Identifica el tipo de encabezado inmediatamente siguiente al encabezado de Ruteo. Usa los mismos valores del campo protocolo de IPv4 [RFC1700].
Long Ext Enc	Entero de 8 bits sin signo. Longitud del encabezado de Ruteo en unidades de 8 bytes, no incluye los primeros 8 bytes. Para el encabezado de Ruteo Tipo 0 Long Ext Enc es igual a dos veces el número de direcciones en el encabezado
Tipo de Ruteo	0
Segmentos Restantes	Entero de 8 bits sin signo. Número de segmentos de ruteo restantes, por ejemplo, el número explícitamente listado de nodos intermedios que todavía tienen que ser visitados antes de alcanzar el destino final.

Reservado	Campo reservado de 32 bits. Inicializado a cero para transmisión, ignorado en recepción
Direcciones [1... n]	Vector de direcciones de 128 bits, numerado de 1 a n

Las direcciones multicast no deben aparecer en un encabezado de Ruteo de tipo 0, o en el campo de Dirección IP Destino de un paquete acarreando un encabezado de Ruteo de Tipo 0.

3.5.5. Encabezado Fragmento

El encabezado Fragmento es usado por una fuente IPv6 para enviar un paquete más grande que lo que podría caber en la ruta MTU a su destino. (Nota: a diferencia de IPv4, la fragmentación en IPv6 es desempeñada solamente en los nodos origen, no por los ruteadores a lo largo de la ruta de entrega del paquete) el encabezado Fragmento es identificado por un valor de Próximo Encabezado de 44 en el encabezado inmediatamente precedente, y tiene el siguiente formato:

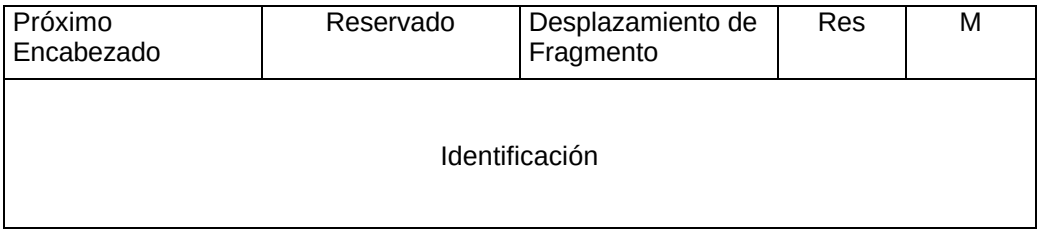

Figura 10. Encabezado Fragmento

Próximo Encabezado	Selector de 8 Bits. Identifica el tipo inicial del encabezado de la parte Fragmentable del paquete original. Usa los mismos valores del campo protocolo de IPv4 [RFC1700 et seq.]
Reservado	Campo de 8 bits reservado. Inicializado a cero para la transmisión; ignorado en la recepción.
Desplazamiento de Fragmento	Entero de 13 bits sin signo. El desplazamiento, en unidades de 8 bytes, de los datos siguientes a este encabezado, relativo al inicio de la parte Fragmentable del paquete original.
Res	Campo de 2 bits reservado. Inicializado a cero para transmisión; ignorado para la recepción.
Bandera M	1 = más fragmentos; 0 = último fragmento
Identificación	32 bits. Se describe a continuación.

En orden para enviar un paquete que es muy grande para caber en la MTU de la ruta a su destino, un nodo Origen puede dividir el paquete en fragmentos y enviar cada fragmento como un paquete separado, para ser reensamblado en el receptor.

Para todos los paquetes que son fragmentados, el nodo Origen genera un valor de Identificación. La Identificación debe ser diferente a cualquier otro paquete fragmentado enviado recientemente[8] con la misma Dirección IP Origen y Dirección IP Destino. Si un encabezado de Ruteo está presente, la Dirección IP Destino de interés es la del destino final.

El paquete inicial, grande, no fragmentado es referido al “paquete original”, y se considera que consiste de dos partes, como se ilustra:

Paquete original:

Figura 11. Ejemplo un paquete original

La parte No Fragmentable consiste del encabezado IPv6 más cualquier encabezado de extensión que debe ser procesado por nodos en la ruta al destino, esto es, todos los encabezados superiores incluyendo el encabezado de Ruteo si está presente, sino el encabezado de Opciones Salto a Salto si esta presente, sino ninguno de los encabezados de extensión.

La parte Fragmentable consiste del resto del paquete, esto es, cualquier encabezado de extensión que necesita ser procesado solamente por el nodo(s) final, más el encabezado de la capa superior y los datos.

La parte Fragmentable del paquete original es dividida en fragmentos, cada uno, excepto posiblemente el último (de más a la derecha), siendo un entero múltiplo de 8 bytes de largo. Los fragmentos son transmitidos en “paquetes fragmento” como se ilustra:

Paquete original

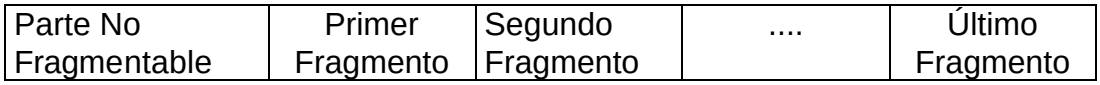

Figura 12. Ejemplo de división del paquete original

[8] Recientemente significa dentro del máximo tiempo de vida de un paquete, incluyendo el tiempo de tránsito del Origen al Destino y el tiempo gastado aguardando el reensamblado con otros fragmentos del mismo paquete. Sin embargo, no es requerido que un nodo origen conozca el máximo tiempo de vida de un paquete. Más bien, es asumido que el requerimiento puede encontrarse manteniendo el valor Identificación como un simple contador "wrap-around" de 32 bits, incrementado cada vez que un paquete debe ser fragmentado. Es una opción de implementación mantener un contador simple para el nodo o múltiples contadores, por ejemplo, uno para cada una de las posibles Direcciones Origen del nodo, o una para cada combinación activa (dirección fuente, Dirección IP Destino).

Paquetes fragmento:

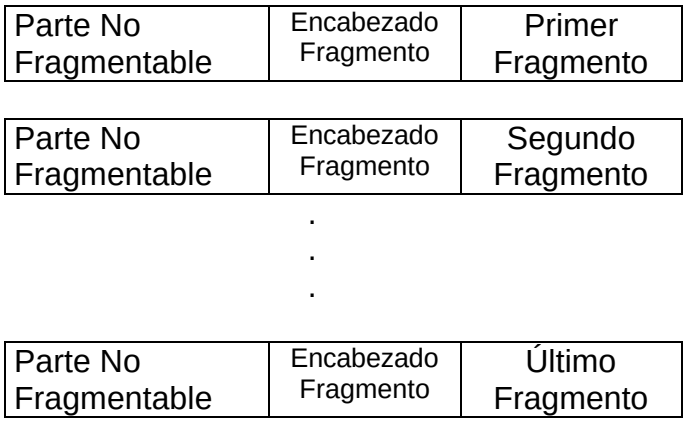

Figura 13. Paquetes fragmento

Cada paquete fragmento está compuesto de:

1. La parte No Fragmentable del paquete original, con la Longitud de Carga Útil del encabezado original cambiada para contener la longitud de este paquete fragmento solamente (excluyendo la longitud del encabezado IPv6 por si mismo), y el campo Próximo Encabezado del último encabezado de la parte No Fragmentable cambiado a 44.

2. Un Encabezado Fragmento contiene:

   El valor del Próximo Encabezado que identifica al primer encabezado de la parte Fragmentable del paquete original.

   Un Desplazamiento de Fragmento conteniendo el desplazamiento, en unidades de 8 bytes, relativas al inicio de la Parte Fragmentable del paquete original. El Desplazamiento del Fragmento del primer fragmento (es del más a la izquierda) es igual a 0.

   Una bandera M tiene el valor de 0 si el fragmento es el último (el de más a la derecha), en caso contrario la bandera M tiene el valor de 1.

   El valor de Identificación generado por el paquete original.

3. El fragmento en sí mismo.

Las longitudes de los fragmentos deben ser escogidas tal que el fragmento resultante quepa dentro del MTU de la(s) ruta(s) destino del (los) paquete(s).

En el destino, los paquetes fragmento son reensamblados dentro del original, en forma no fragmentada, como se ilustra:

Paquete original reensamblado

Parte No Fragmentable	Parte Fragmentable

Figura 14. Paquete original reensamblado

Las siguientes reglas gobiernan en el reensamblado:

Un paquete original es reensamblado sólo de los paquetes fragmento que tienen la misma Dirección Fuente, Dirección IP Destino e Identificación de Fragmento.

La parte No Fragmentable del paquete reensamblado consiste de todos los encabezados superiores, pero no incluyendo, el encabezado Fragmento del primer paquete fragmento (esto es, los paquetes cuyo Desplazamiento de Fragmento es igual a cero), con los siguientes dos cambios:

El campo Próximo Encabezado del último encabezado de la parte No Fragmentable es obtenida del campo Próximo Encabezado del campo Próximo Encabezado del primer encabezado Fragmento.

La Longitud de Carga Útil del paquete reensamblado es calculada de la longitud de la parte No Fragmentable y la longitud y el desplazamiento del último fragmento. Por ejemplo, una formula para calcular la Longitud de Carga Útil del paquete original reensamblado es:

LCU.orig = LCU.inicio – LF.inicio – 8 + (8 * DF.fin) + LF.fin

Donde

LCU.orig	=	Campo de Longitud de Carga Útil del paquete reensamblado
LCU.inicio	=	Campo de Longitud de Carga Útil del primer paquete fragmento
LF.inicio	=	Longitud del fragmento siguiente al encabezado Fragmento del primer paquete fragmento
DF.fin	=	Campo Desplazamiento del Fragmento del encabezado Fragmento del último paquete fragmento
LF.fin	=	Longitud del fragmento siguiente al encabezado Fragmento del último paquete fragmento

La parte Fragmentable del paquete reensamblado es construida de los fragmentos siguientes de los encabezados Fragmento en cada uno de los paquetes fragmento. La longitud de cada fragmento es computada o calculada restando de la Longitud de Carga Útil del Paquete la longitud de los encabezados entre el encabezado IPv6 y el fragmento en si mismo; su posición relativa en la parte Fragmentable es calculada de su valor de Desplazamiento de Fragmento.

El encabezado Fragmento no esta presente en el reensamblado del paquete final.

Las siguientes condiciones de error pueden surgir cuando se están reensamblado paquetes fragmentados:

Si son recibidos paquetes insuficientes para completar el reensamblado de un paquete dentro de los primeros 60 segundos de la recepción del primer fragmento de ese paquete, el reensamblado de ese paquete debe ser abandonado y todos los fragmentos que han sido recibidos de ese paquete deben ser descartados. Si el primer fragmento (por ejemplo, el primero con un Desplazamiento de Fragmento de cero) ha sido recibido, un mensaje ICMP de Tiempo Excedido en el Reensamblado de Fragmentos debe ser enviado al Origen de ese fragmento.

Si la longitud de un fragmento, tal como se dedujo del campo de Longitud de Carga Útil del fragmento, no es múltiplo de 8 bytes y la bandera M del fragmento es igual a 1, entonces el fragmento debe ser descartado y un mensaje ICMP de Problema de Parámetro, código 0, debe ser enviado al Origen del fragmento, señalando al campo de la Longitud de Carga Útil del paquete Fragmento.

Si la longitud y desplazamiento de un fragmento son tales que la Longitud de Carga del paquete reensamblado del fragmento excediera de 65535 bytes, entonces el fragmento debe ser descartado y un mensaje ICMP de Problema de Parámetro, código 0, debe ser enviado al Origen del fragmento, señalando al campo Desplazamiento de Fragmento del paquete fragmento.

Las siguientes condiciones no se espera que ocurran, pero no son consideradas errores si lo hacen:

El número y el contenido de los encabezados precedentes al encabezado Fragmento de diferentes fragmentos del mismo paquete original pueden diferir. Cualquiera de los encabezados que están presentes, precediendo al fragmento Encabezado en cada paquete fragmento, son procesados cuando los paquetes arriban, previamente que los paquetes se encolen para el reensamblaje. Sólo esos encabezados en el paquete con Desplazamiento cero son retenidos en el paquete reensamblado.

Los valores de Próximo Encabezado en los encabezados Fragmento de diferentes fragmentos del mismo paquete original pueden diferir. Sólo el paquete fragmento con Desplazamiento cero es usado para reensamblado.

3.5.6. Encabezado de Opciones de Destino

El encabezado de Opciones de Destino es usado para soportar información adicional que debe ser examinada solamente por el(los) nodo(s) destino. El encabezado de Opciones de Destino es identificado por un valor de Próximo Encabezado igual a 60 en el encabezado inmediatamente precedente, y tiene el siguiente formato:

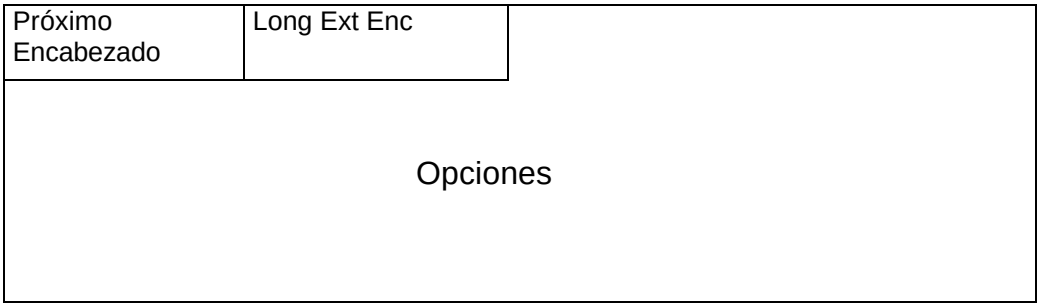

Figura 15. Encabezado de Opciones de Destino

Próximo Encabezado	Selector de 8 Bits. Identifica el tipo de encabezado inmediatamente siguiente al encabezado de Opciones de Destino. Usa los mismos valores del campo protocolo de IPv4 [RFC1700].
Long Ext Enc	Entero de 8 bits sin signo. Longitud del encabezado de Opciones de Destino en unidades de 8 bytes, no incluye los primeros 8 bytes.
Opciones	Campo de longitud variable, de longitud tal que el encabezado de Opciones Salto a Salto es un múltiplo entero de 8 bytes de largo. Contiene una o mas opciones TLV codificadas.

Note que hay dos maneras posibles de codificar la información opcional de destino en un paquete IPv6: como una opción en encabezado de Opciones de Destino, o como un encabezado de extensión separado. El encabezado Fragmento y el encabezado de Autenticación son ejemplos del anterior método. El método usado depende de la acción deseada por el nodo destino que no comprenda la información opcional:

- Si la acción deseada es para que el nodo destino descarte el paquete y, sólo si la Dirección IP Destino de Paquete no es una dirección Multicast, envíe un mensaje ICMP de Tipo No reconocido a la Dirección IP Origen del paquete, entonces la información puede ser codificada como un

encabezado separado o como una opción en el encabezado de Opciones de Destino cuyo Tipo de Opción tiene el valor 11 en su dos bits de más alto orden. La selección puede depender de factores como cuál toma menos bytes o cuál tiene una mejor alineación o un análisis más eficiente.

- Si cualquier otra acción es deseada, la información debe ser codificada como una opción en el encabezado de Opciones de Destino cuyo Tipo de Opción tiene el valor 00, 01 ó 10 en sus dos bits de más alto orden, especificando la acción deseada.

3.5.7. Encabezado No hay Siguiente

El valor 59 en el campo de Próximo Encabezado de un encabezado IPv6 o de cualquier encabezado de extensión indica que no hay nada detrás de dicho encabezado. Si el campo de Longitud de Carga Útil del encabezado IPv6 indica la presencia de bytes pasados el fin de un encabezado el cual el campo de Próximo Encabezado contiene un 59, esos bytes deben ser ignorados, y pasados sin cambio si el paquete es reenviado.

3.6. Arquitectura de direccionamiento de IPv6.

3.6.1. Direccionamiento de IPv6.

Las direcciones en IPv6 son identificadores de 128 bits para interfaces[9] y conjuntos de interfaces. Existen tres tipos de direcciones:

- Unicast: Un identificador para una interfaz simple. Un paquete enviado a una dirección unicast es entregado a la interfaz identificado por la dirección.

- Anycast: Un identificador para un conjunto de interfaces (típicamente perteneciendo a diferentes nodos). Un paquete enviado a una dirección anycast es entregado a una de las interfaces identificadas por la dirección (a la más "cercana", de acuerdo a la medida de distancia del protocolo de ruteo).

- Multicast: Un identificador para un conjunto de interfaces (típicamente perteneciendo a diferentes nodos). Un paquete que es enviado a una dirección multicast es entregado a todas las interfaces identificadas por la dirección.

- No existen las direcciones de broadcast en IPv6, su función es reemplazada por las direcciones multicast.

[9] Por interfaces e interfaces físicas se refiere a las tarjetas de red o al dispositivo por el cual se conectan a una red.

En IPv6, todos los ceros y todos los unos son valores legales para cualquier campo, a menos que sean específicamente excluidos. Específicamente, los prefijos pueden contener campos con valor cero o finalizar en cero.

3.6.2. Modelo de direccionamiento

Las direcciones IPv6 de todos los tipos son asignados a interfaces, no a nodos. Una dirección unicast IPv6 se refiere a una interfaz simple. Desde que cada interfaz pertenece a un nodo simple, cualquiera de las direcciones unicast de las interfaces del nodo puede ser usado como un identificador para el nodo.

Todas las interfaces requieren tener al menos una dirección unicast de "red local". Una interfaz simple puede también ser asignada a múltiples direcciones IPv6 de cualquier tipo (unicast, anycast, multicast) o alcance. Las direcciones unicast con alcance más grande que el "alcance de la red" no son exigidas por interfaces que no son usadas como el origen o el destino de cualquier paquete IPv6 o por no vecinos. Esto, algunas veces es conveniente para interfaces punto a punto. Hay una excepción a este modelo:

> Una dirección unicast de un conjunto de direcciones unicast puede ser asignado a múltiples interfaces físicas si la implementación trata las múltiples interfaces físicas como una interfaz cuando es presentada a la capa de Internet. Esto es útil para el compartimiento de carga sobre múltiples interfaces físicas.

Actualmente IPv6 continúa el modelo de IPv4 en el que un prefijo de subred es asociada con un enlace. Múltiples prefijos de subred pueden ser asignados al mismo enlace.

3.6.3. Representación en texto de las direcciones

Hay tres formas convencionales para representar direcciones IPv6 como cadenas de texto:

1. La forma preferida es x:x:x:x:x:x:x:x, donde las x son los valores hexadecimales de las ocho piezas de 16 bits de la dirección. Por ejemplo:

   FEDC:BA98:7654:3210:FEDC:BA98:7654:3210

   1080:0:0:0:8:800:200C:417A

   Note que no es necesario escribir los ceros restantes en un campo individual, pero debe haber al menos un número en todos los campos.

2. Debido a que algunos métodos de asignación de ciertos estilos de direcciones de IPv6 será común para direcciones contener largas cadenas de bits en cero. Para facilitar la escritura de direcciones conteniendo bits en

cero, una sintaxis especial está disponible para comprimir los ceros. El uso de "::"[10] indica múltiples grupos de ceros de 16 bits. El "::" puede aparecer solamente una vez en la dirección. El "::" puede también ser usado para comprimir los espacios y/o arrastrar ceros en una dirección.

Por ejemplo las siguientes direcciones:

1080:0:0:0:8:800:200C:417A	una dirección unicast
FF01:0:0:0:0:0:0:101	una dirección multicast
0:0:0:0:0:0:0:1	la dirección loopback
0:0:0:0:0:0:0:0	las direcciones no especificadas

pueden ser representadas como:

1080::8:800:200C:417A	una dirección unicast
FF01::101	una dirección multicast
::1	la dirección loopback
::	las direcciones no especificadas

3. Una forma alternativa que algunas veces es más conveniente cuando se negocia con un ambiente mezclado de nodos con IPv4 e IPv6 es x:x:x:x:x:x:d.d.d.d, donde las 'x' son los valores hexadecimales de las seis piezas de 16 bits de alto orden, y las 'd' son los valores decimales de las cuatro piezas de 8 bits de bajo orden de las direcciones (representación estándar de IPv4). Ejemplos:

   0:0:0:0:0:0:13.1.68.3

   0:0:0:0:0:FFFF:129.144.52.38

   o en la forma comprimida:

   ::13.1.68.3

   ::FFFF:129.144.52.38

3.6.4. Representación en texto de prefijos de direcciones

La representación en texto de los prefijos de las direcciones de IPv6 es similar a la forma de representación de los prefijos en IPv4 en la notación CIDR.

Un prefijo de dirección IPv6 es representado por la notación:

dirección_ipv6/longitud_del_prefijo

donde

[10] Abreviado colon hex

dirección_ipv6	es una dirección IPv6 en cualquiera de las notaciones listadas
longitud_del_prefijo	es un valor decimal especificando cuántos de los bits contiguos más a la izquierda de la dirección comprenden el prefijo

Por ejemplo, las siguientes son representaciones legales del prefijo de 60 bits 12AB00000000CD3 (hexadecimal):

12AB:0000:0000:CD30:0000:0000:0000:0000/60
12AB::CD30:0:0:0:0/60
12AB:0:0:CD30::/60

Las siguientes no son representaciones legales del prefijo anterior:

12AB:0:0:CD3/60	Se pueden cortar ceros, pero no arrastrarlos dentro de cualquier pedazo de 16 bits de la dirección
12AB::CD30/60	la dirección a la izquierda de "/" se expande a 12AB:0000:0000:0000:0000:000:0000:CD30
12AB::CD3/60	la dirección a la izquierda de "/" se expande a 12AB:0000:0000:0000:0000:000:0000:0CD3

Cuando se escriben una dirección de un nodo y un prefijo de la dirección del nodo, las dos pueden ser combinadas como sigue:

la dirección del nodo	12AB:0:0:CD30:123:4567:89AB:CDEF
y su número de subred	12AB:0:0:CD30::/60

puede ser abreviada como

12AB:0:0:CD30:123:4567:89AB:CDEF/60

3.7. Representación del tipo de dirección

El tipo especifico de una dirección IPv6 está indicado por los bits guía en la dirección. El campo comprende esos bits guía llamados el Prefijo Formato (FP). La asignación inicial de dichos prefijos es como sigue:

Asignación	Prefijo (binario)	Fracción del espacio de direcciones
Reservada	0000 0000	1/256
No asignada	0000 0001	1/256
Reservada para Asignación NSAP	0000 001	1/128
Reservada para Asignación IPX	0000 010	1/128
No asignada	0000 011	1/128
No asignada	0000 1	1/32
No asignada	0001	1/16
Direcciones Globales Unicast Agregables	001	1/8
No asignada	010	1/8
No asignada	011	1/8
No asignada	100	1/8
No asignada	101	1/8
No asignada	110	1/8
No asignada	1110	1/16
No asignada	1111 0	1/32
No asignada	1111 10	1/64
No asignada	1111 110	1/128
No asignada	1111 1110 0	1/512
Dirección Unicast Link-Local	1111 1110 10	1/1024
Dirección Unicast Site-Local	1111 1110 11	1/1024
Direcciones Multicast	1111 1111	1/256

Esta asignación soporta la asignación directa de direcciones de agregación, direcciones de uso local, y direcciones multicast. El espacio es reservado para direcciones NSAP y direcciones IPX. El resto del espacio de direcciones no está asignado para uso fututo. Esto puede ser usado para la expansión del uso existente (por ejemplo, direcciones agregables adicionales, etc.) o nuevos usos (por ejemplo, separar localizadores e identificadores). El 15% del espacio de direcciones esta inicialmente asignado. El 85% restante está reservado para un uso futuro.

Las direcciones unicast son distinguidas de las direcciones multicast por el valor del byte de alto orden de las direcciones: un valor de FF (11111111) identifica una dirección como una dirección multicast; cualquier otro valor identifica una dirección como una dirección unicast. Las direcciones anycast son tomadas del espacio de direcciones de unicast, y no son sintácticamente distinguibles de las direcciones unicast.

3.7.1. Direcciones unicast.

Las direcciones unicast de IPv6 son agregables con mascaras contiguas similares a IPv4 poco entendidas bajo CIDR.

Hay muchas formas de asignación de direcciones unicast en IPv6, incluyendo las direcciones globales agregables unicast, las direcciones NSAP, las direcciones jerárquicas IPX, las direcciones de sitios locales, las direcciones de redes locales, y las direcciones de hosts con capacidad de IPv4. Tipos adicionales de direcciones pueden ser definidos en el futuro.

Los nodos IPv6 pueden tener considerable o poco conocimiento de la estructura interna de la dirección de IPv6, dependiendo del rol que el nodo juegue (por ejemplo, un host en contra de un router). Como mínimo, un nodo puede considerar que las direcciones unicast (incluyendo la propia) tienen la estructura no interna como sigue:

128 bits
Dirección del nodo

Figura 16. Estructura mínima de una dirección unicast

Un host un poco más sofisticado (pero todavía muy simple) puede adicionalmente tener conciencia del (los) prefijo(s) de subred de la(s) liga(s) a las que viene pegado, donde diferentes direcciones pueden tener diferentes valores de n:

n bits	128-n bits
Prefijo de subred	ID de la interfaz

Figura 17. Estructura mínima de una dirección unicast con prefijo de subred

Aun hosts más sofisticados pueden tener conocimiento de otros límites jerárquicos en las direcciones unicast. Aunque un router muy simple puede no tener conocimiento de la estructura interna de las direcciones IPv6 unicast, los ruteadores tendrán generalmente conocimiento de uno o más de los límites jerárquicos para la operación de protocolos de ruteo. Los límites conocidos diferirán de router a router, dependiendo que posiciones controla el router en la jerarquía de ruteo.

Se han definido dos tipos de direcciones unicast de uso local: Link-Local [11] y Site-Local.[12] Las direcciones locales de enlace han sido diseñadas para direccionar un único enlace para propósitos de auto configuración, descubrimiento de vecinos, o situaciones en las que no hay routers. Así, los routers no pueden

[11] Local de Enlace
[12] Local de Sitio

transmitir ningún tipo de paquete con direcciones Site-Local (su ámbito esta limitado a la red local). Se trata de direcciones fe80::<ID de Interfaz>/10.

Las direcciones Site-Local permiten direccionar dentro de un sitio local u organización, sin la necesidad de un prefijo global. Se configuran mediante un identificador de subred de 16 bits. Los ruteadores no deben retransmitir fuera del sitio ningún paquete cuya dirección fuente o destino sea Link-Local (su ámbito está limitado a la red local o de organización). Se trata de direcciones fec0::<ID de interfaz de red>/10.

3.7.2. Identificadores de Interfaz

Los identificadores de interfaz en las direcciones unicast de IPv6 son usados para identificar interfaces en un enlace. Son requeridas ser únicas en ese enlace. Pueden ser únicas en un ámbito de gran alcance. En muchos casos un identificador de interfaz será el mismo que las direcciones de la interfaz en la capa de enlace. El mismo identificador puede ser usado en múltiples interfaces o en un nodo simple.

3.7.3. Las direcciones no especificadas

La dirección 0:0:0:0:0:0:0:0 es llamada la dirección no especificada. No debe ser asignada a ningún nodo. Indica la ausencia de una dirección. Un ejemplo de su uso es el campo Dirección Origen de cualquier paquete IPv6 enviado por un host iniciado antes de que aprenda su propia dirección.

La dirección no especificada no debe ser usada como la Dirección Destino de paquetes IPv6 o en Encabezados de Ruteo de IPv6.

3.7.4. La dirección de lazo cerrado (loopback)

La dirección unicast 0:0:0:0:0:0:0:1 es llamada la dirección de loopback. Puede ser usada por un nodo para enviar un paquete IPv6 a sí misma. No puede ser asignada físicamente a ninguna interfaz. Puede pensarse como asociado con una interfaz virtual.

La dirección loopback no debe ser usada como Dirección Origen en paquetes IPv6 que son enviados fuera del nodo. Un paquete IPv6 con una Dirección Destino loopback no debe nunca ser enviado fuera de un nodo simple y nunca debe ser reenviado por un router IPv6.

3.7.5. Direcciones IPv6 con direcciones IPv4 incrustadas

El mecanismo de transición de IPv6 [TRAN] incluye una técnica para que hosts y ruteadores tuneleen dinámicamente paquetes IPv6 sobre infraestructura de ruteo de IPv4. A los nodos IPv6 que utilizan esta técnica les son asignadas direcciones especiales unicast de IPv6 que pueden acarrear una dirección IPv4 en

los 32 bits de más bajo orden. Este tipo de dirección es denominada una "dirección IPv4 compatible con IPv6" y tiene el formato:

80 bits	16	32 bits
0000……0000	0000	Dirección IPv4

Figura 18. Dirección IPv4 compatible con IPv6

Un segundo tipo, también definido, de dirección IPv6 es la que controla una dirección IPv4 incrustada. Esta dirección es usada para representar la dirección de nodos de IPv4 (aquellos que no soportan IPv6) como direcciones de IPv6. este tipo de dirección se denomina una "dirección IPv4 mapeada en IPv6" y tiene el siguiente formato:

80 bits	16	32 bits
0000……0000	FFFF	Dirección IPv4

Figura 19. Dirección IPv4 mapeada en IPv6

3.7.6. Direcciones Anycast

Una dirección anycast es una dirección que es asignada a más de una interfaz (típicamente pertenecientes a diferentes nodos), con la propiedad de que un paquete enviado a una dirección anycast es ruteado a la interfaz con la dirección "más cercana", de acuerdo a la medida de distancia de los protocolos de ruteo.

Las direcciones anycast son asignadas del espacio de direcciones unicast, usando cualquiera de los formatos definidos de direcciones unicast. Así, las direcciones anycast son sintácticamente indistinguibles de las direcciones unicast. Cuando una dirección unicast es asignada a más de una interfaz, así girándola en una dirección anycast, el nodo al cual le es asignada la dirección debe ser explícitamente configurado para saber que esa es una dirección anycast.

3.7.7. Dirección Multicast

Una dirección multicast es un identificador para un grupo de nodos. Un nodo puede pertenecer a cualquier número de grupos multicast. Las direcciones multicast tienen el siguiente formato:

8	4	4	112 bits
11111111	banderas	alcance	ID de grupo

Figura 20. Dirección multicast

- 11111111 en el inicio de la dirección identifican la dirección como una dirección multicast

- banderas es un conjunto de 4 banderas:

0	0	0	T

  Las tres banderas de mas alto orden están reservadas, y deben ser inicializadas a 0.

  T=0 indica una dirección multicast asignada permanentemente (adecuadamente conocida), asignada por la autoridad global de numeración de Internet.

  T=1 indica una dirección multicast asignada temporalmente (“transitoria”)

- Alcance es un valor de alcance de 4 bits usado para limitar el alcance del grupo multicast. Los valores son:

0	Reservada
1	Nodo de alcance local
2	Enlace de alcance local
3	(no asignada)
4	(no asignada)
5	Sitio de alcance local
6	(no asignada)
7	(no asignada)
8	Organización de alcance local
9	(no asignada)
A	(no asignada)
B	(no asignada)
C	(no asignada)
D	(no asignada)
E	Alcance global
F	Reservada

- ID de grupo identifica el grupo multicast, permanente o transitorio, dentro del alcance dado.

Capítulo 4. Implementación

Introducción.

En este capitulo se muestra como configurar un servidor corriendo Linux PPP 6.4 con un Kernel 2.2.x[1] basado en Linux Red Hat 6.2. A diferencia de los anteriores capítulos, éste intenta ser un pequeño manual para instalar y configurar el Kernel y las herramientas básicas para un servidor corriendo IPv6.

4.1. Selección del Kernel y del software básico

La selección del Kernel se refiere a una posible actualización, como se menciono en el capitulo 3, por cuestiones de seguridad, también es recomendable porque un Kernel nuevo puede tener soporte para dispositivos nuevos, como puertos USB, scanners o CDRW, etc.

La recomendación de actualizar el Kernel se hace para usuarios expertos, ya que aparte de configurar el soporte para IPv6, se tiene que conocer todas las opciones del Kernel que deben estar habilitadas para que el sistema funcione. Si se realiza se debe revisar el “Kernel How To” y verificar que todas las opciones del Kernel anterior estén seleccionadas.

En caso de usar una versión de Linux mas reciente (distribuciones basadas en el Kernel 2.4.x, Red Hat 7.2 por ejemplo), no es necesario recompilar, el soporte para IPv6 viene incluido por defecto, solo es necesario verificar el funcionamiento de los demás paquetes. Si existe algún cambio o modificación que sea necesario realizar, se indicará pertinentemente.

Para una mayor comprensión, las instrucciones o resultados de comandos se presentan con *letra cursiva*.

4.2. Cómo configurar un Kernel con soporte para IPv6

Se supone se está utilizando un Kernel 2.2.x. Utilizar un Kernel mas reciente es decisión del administrador y este capítulo no lo cubre.

[1] La x se relaciona con la versión del Kernel, la última versión de la serie 2.2.x fue la 2.2.19.

Primero, se debe accesar a la máquina como root[2], si no se cuenta con el acceso, comunicarse con el administrador del sistema para que se lo proporcione. Después se debe hacer un disco de arranque si no se creó uno en la instalación. Para crea un disco se puede escribir:

mkbootdisk `uname -r`

esta instrucción crea una imagen del Kernel para poder arrancar desde el disquete de 31/2 en caso de que el LILO[3] resulte dañado o con alteraciones y no se pueda acceder al sistema operativo.

4.3. Configuración del Kernel

Si el sistema operativo es lo que evita que una computadora sea un montón de plásticos y circuitos inútiles, el Kernel es lo que hace que el sistema operativo funcione. El Kernel es el gestor entre las interfaces de usuario (shell) y el sistema operativo, esto incluye la capacidad de gestionar la memoria, las tarjetas de red, etc.

Es más cómodo y funcional tener el sistema X Window corriendo en el equipo. De no ser así, no existe problema alguno, sólo que será un poco más complicado.

Primero se debe lograr acceso a la maquina como root, en una consola teclear lo siguiente:

cd /usr/src/linux/

esto nos lleva al directorio donde se encuentra el código fuente[4] del Kernel[5] y una vez ahí escribir

make menuconfig, ver figura 1.

seleccionar “Networking Options”, aquí es donde se encuentran las opciones que se deben marcar para habilitar el Kernel para que maneje el protocolo IPv6.

---

[2] SúperUsuario
[3] Linux Loader – Cargador de Linux
[4] Ya que el Kernel de Linux también esta bajo los términos de la licencia GNU todo el código fuente del mismo se encuentra presente.
[5] El código fuente se puede instalar desde la opción “Kernel Hacking” en el proceso de instalación de Linux Red Hat.

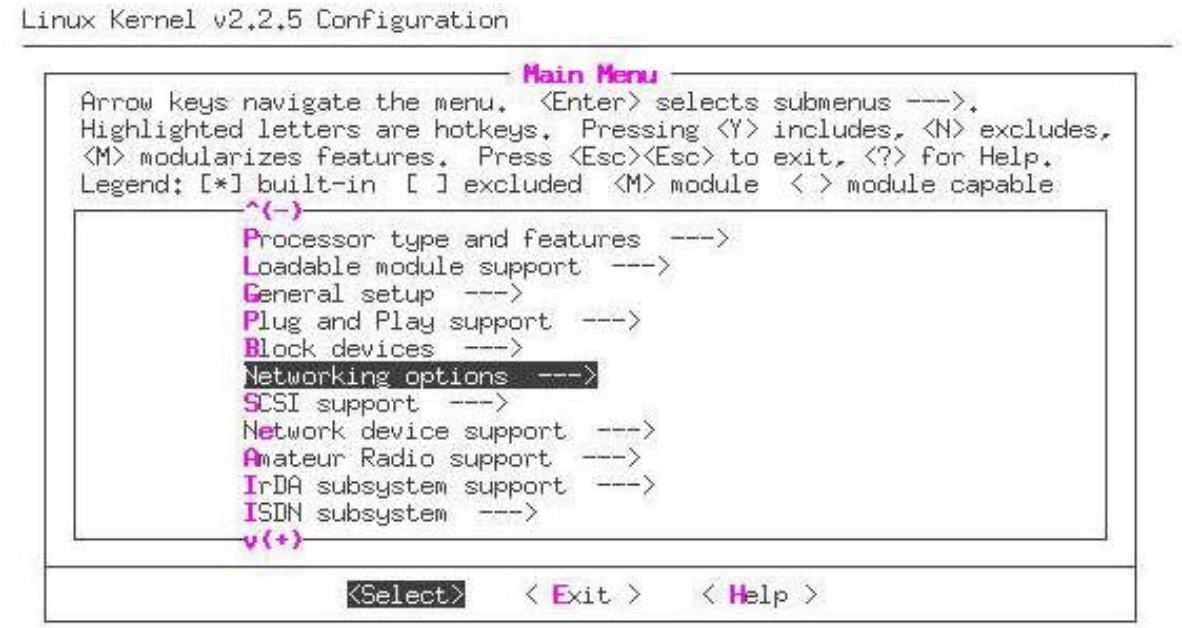

Figura 1. Pantalla del menú principal del menuconfig

Seleccionar "The IPv6 protocol", "IPv6: enable EUI-64 token format", "IPv6: disable provider based addresses" ver figura 2.

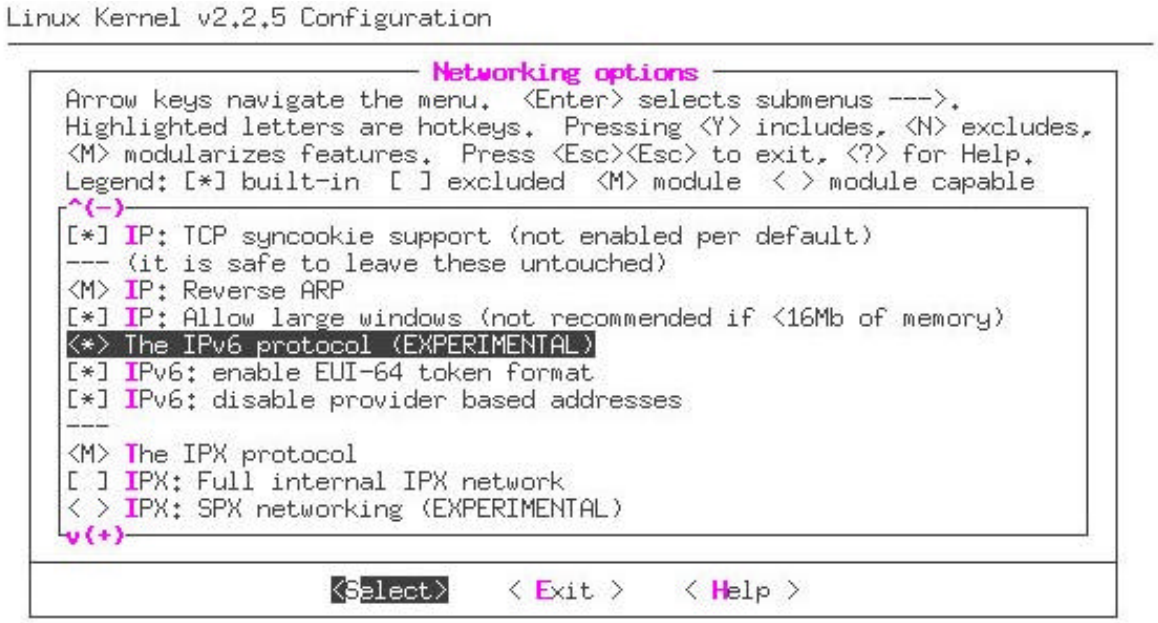

Figura 2. Pantalla de la selección de las opciones de configuración del Kernel para IPv6

Para grabar los cambios realizados al sistema es necesario seleccionar dos veces “exit”, la primera para regresar al menú principal y la segunda para salir del programa, se responde “yes” a la siguiente pregunta “Do you wish to save your new Kernel configuration?". [6]

En este momento se escribe una secuencia de comandos para recompilar el Kernel y dejarlo listo para instalarse

1. *make dep*

2. *make clean*

3. *make bzImage*

4. *make modules*

Ahora se explica brevemente lo que quiere decir cada comando

1. Esta instrucción establece las dependencias correctamente. Prepara los enlaces con los archivos que le hacen falta.

2. Esta instrucción lo que hace es eliminar ficheros y objetos de versiones anteriores del núcleo, lo que nos ayudará a recuperar espacio en el disco duro.

3. Esta instrucción es la que genera la imagen de núcleo, es el paso en el que realmente se compila el núcleo.

4. Esta instrucción carga las nuevas opciones que fueron agregadas como módulos en la configuración del Kernel.

Si todo compila bien, esto quiere decir que el proceso de instalación se desarrolla sin mostrar errores y se genera el archivo bzImage (nueva imagen del Kernel) en /usr/src/linux/arch/i386/boot, se ejecuta la siguiente instrucción

make modules_install

Esta instrucción generara todos los directorios e instala los nuevos módulos

En caso de error se ejecuta la siguiente instrucción

make rmproper

[6] “¿Desea grabar la nueva configuración del Kernel?”

Esta instrucción elimina las dependencias que se crean a los módulos al ejecutar las instrucciones anteriores, ahora se debe reiniciar el proceso cuidando que el orden sea el correcto.

Ahora que ya esta compilado el Kernel sin errores, es necesario copiar la nueva imagen, entonces

cp /arch/i386/boot/bzImage /boot/vmlinuz-2.2.x-ipv6

Hasta este punto tenemos el Kernel compilado y los módulos, falta agregar el nuevo Kernel al LILO, se tiene que editar el archivo lilo.conf que se encuentra en el directorio /etc.

vi /etc/lilo.conf

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
image=/boot/vmlinuz-2.2.5-22-ipv6
        label=linux
        root=/dev/hda1
        read-only
image=/boot/vmlinuz-2.2.5-22
        label=linux-ipv4
        root=/dev/hda1
        read-only
~
~
~
~
~
~
~
~
~
~
~
"/etc/lilo.conf" 13 lines, 215 characters
```

Figura 3. Pantalla del archivo de configuración /etc/lilo.conf para agregar la nueva imagen del Kernel.

Esta es una configuración de ejemplo, pero puede servir, la idea es agregar la imagen del Kernel para que el LILO sepa de él y lo tome en cuenta, se puede copiar la entrada del Kernel anterior (el que no está compilado para IPv6) y sólo cambiarle el nombre a la línea del vmlinuz y a la de label. Después, se debe ejecutar

/sbin/lilo

para actualizar los cambios. Ya esta listo para reiniciar, para seleccionar el nuevo Kernel, al momento de aparecer el prompt del LILO escribir lo que se puso en la línea label.

Si todo sale bien, se debe ver algo como esto cuando inicia

IPv6 v0.8 for NET4.0
IPv6 over IPv4 tunneling driver

Entonces, el Kernel ya está habilitado para funcionar bajo IPv6 y falta instalar y configurar el software restante.

4.4. Instalación del software

Todo el software que se menciona en esta sección se descargó de Internet, aquí se proporcionan los URLS para descargarlos, si alguna versión no corresponde con la mostrada aquí, se debe descargar la última disponible, se debe descargar software que sea capaz de manejar direcciones de IPv6.

4.4.1. Configuración de las herramientas de red.

Muchas de las herramientas de red necesarias sólo soportan IPv4, pero para la configuración de la Interfaz se necesitan las herramientas habilitadas para IPv6.

Para saber si las herramientas de manejo de red se ocupa el comando netstat para verificar que la opción que se encuentra instalada en el servidor esta lista.

netstat – ?

si la salida del comando muestra un enunciado similar al siguiente

List of possible address families (which support routing):
inet (DARPA Internet) ***inet6 (IPv6)*** *ax25 (AMPR AX.25)*
netrom (AMPR NET/ROM) ipx (Novell IPX) ddp (Appletalk DDP)

la sección de Configuración de las herramientas de red puede ser omitida, es decir, el servidor tiene las herramientas de red necesarias para manejar IPv6.

Si no es así, el paquete puede ser descargado de la siguiente URL: http://www.tazenda.demon.co.uk/phil/net-tools/

El nombre y versión del paquete net-tools-1.54.tar.gz. Net-Tools debe ser construido (instalado, compilado) con un Kernel (2.2.x, 2.4.x) habilitado para IPv6

Aplicación	Ruta	Descripción
Hostname	/bin/hostname	
Netstat	/bin/netstat	Status de la red
Arp	/sbin/arp	Manipulación del cache ARP
Ifconfig	/sbin/ifconfig	Configuración de la Interfaz
Rarp	/sbin/rarp	Manipulación del cache RARP
Route	/sbin/route	Manipulación del Internet route

Tabla 1. Lista de las aplicaciones que se deben actualizar, ruta en que se encuentran y descripción

Desempacado, configuración e instalación.

1. Cambiarse dentro del directorio fuente

   cd /usr/src

2. Desempaquetar el nuevo paquete

   tar xzf tu-path/vet-tools-versión.tar.gz –C /usr/src

3. Renombrar el nuevo directorio

   mv net-tools net-tools-versión

4. Ahora, crear/sobreescribir la liga simbólica, necesaria para los paths más cortos

   ln -sf /usr/src/net-tools-version /usr/src/net-tools

5. Cambiarse al directorio fuente:

   cd /usr/src/net-tools

6. Configurar las opciones de compilación: make; make clean, make config. ¡No debe olvidar marcar otras opciones dependiendo del sistema!

Opciones de Net-tools	Sub opción	Selección
GNU gettext		si, si glibc-2
Familias de protocolos	Familia de protocolos UNIX	Si
	Familia de protocolos INET (TCP/IP)	Si
	Familia de protocolos INET6 (IPv6)	Si
Tipos de dispositivos de Hardware	Soporte SIT (IPv4 en IPv6)	Si

Tabla 2. Opciones mínimas que se deben marcar para el manejo de las herramientas de red

7. Compilar

   make (no debe haber errores!)

8. Instalar

   make install

Los binarios y los manuales son copiados directamente en los directorios dados, no se debe preocupar por un mensaje “"ls: *.[158]: directory not found", es causado por que no existen las páginas man multilengua (la versión en Ingles de EU siempre será instalada).

Esta herramienta es mantenida por philb@gnu.org

4.4.2. Utilerías de IP.

En sí, el paquete iputils contiene el paquete ping, el cual es necesario para verificar si redes o servidores responden. Por lo tanto, se debe verificar si la versión instalada está lista

rpm -q --qf "%{NAME}-%{VERSION}\n" iputils

si el resultado es igual a *iputils-20000121* o superior, esta sección puede ser omitida.

El URL para descargarlo

ftp://ftp.inr.ac.ru/ip-routing/

Red Hat 6.2 contiene iputils-20000121, el cual está habilitado para IPv6. Por lo tanto, este es uno de los paquetes que no necesitan ser actualizados a menos que se este usando una versión anterior.

Este paquete es mantenido por kuznet@ms2.inr.ac.ru

4.4.3. xinetd (eXtended InterNET super Daemon)

xinetd es un reemplazo para el inetd, el demonio de servicios de Internet. Las funciones que el inetd realiza son la asignación de servicios de comunicaciones, escucha las conexiones en los sockets, y cuando hay alguna petición invoca al programa que presta el servicio. Por ejemplo, en lugar de correr permanentemente telnetd, este es 'despertado' por inetd cuando alguien 'golpea la puerta'. xinetd es una implementación muy mejorada de inetd (mas seguro, mas estable y responde mucho mejor cuando concurren muchas conexiones).

Normalmente, el súper demonio de Internet no está listo para manejar direcciones de IPv6, y no todos los demonios deben correr en modo “stand-alone”.

El URL del paquete es:

Sitio principal: http://synack.net/xinetd/

Sitios de RPM: ftp://ftp.freshmeat.net/pub/rpms/xinetd/

Ambos compilados sin soporte para IPv6. Ya que se tienen los paquetes, lo siguiente es desempacarlo, configurarlo e instalarlo.

Desempacar, configurar e instalar

1. Cambiarse dentro del directorio de fuentes

   cd /usr/src

2. Desempacar la nueva fuente

   tar xvz tu_path/xinet-version.tar.gz

3. Cambiarse al directorio fuente

   cd xinet-version

4. Configurar

Si se quiere colocar los binarios en /usr/*

./configure --with-inet6 --prefix=/usr

Si se quiere colocar los binarios en /usr/local/*

./configure --with-inet6 --prefix=/usr/local

Si se quiere colocar los binarios en /usr/inet6/*

./configure --with-inet6 --prefix=/usr/inet6

5. Compilar

   make clean
   make

   ¡No debe haber errores!

6. Instalar
   make install

Los manuales y binarios son copiados dentro del directorio dependiendo de lo especificado en configuración.

Crear un archivo de configuración del anterior inetd.conf

cat /etc/inetd.conf | /path_configure/itox -daemon_dir /usr/sbin/tcpd >/etc/xinetd.conf

Editar /etc/xinetd.conf y adecuarlo a los requerimientos del sistema.

Para sistemas a partir del Kernel 2.4.x el xined es el Súper Demonio de Internet por defecto para dichos sistemas. Para Red Hat Linux 7.2 no es necesario reinstalar un nuevo paquete, solamente se modifican los siguientes archivos para que el xinetd tome en cuenta el manejo de direcciones de IPv6.

- Cambiar/agregar en **/etc/sysconfig/network**

  NETWORKING_IPV6=yes # Habilita la inicialización global de IPv6

- Cambiar/agregar en **/etc/sysconfig/network-scripts/ifcfg-device**

  IPV6INIT="yes" # Habilita la inicialización de IPv6 para esta interfaz

4.4.3.1. Pruebas con IPv6

Habilitar un servicio interno (por ejemplo daytime) en el archivo de configuración, (re)arrancar el demonio de la siguiente manera:

 # cd path_de_configure_xinetd/xinetd -f /etc/xinetd.conf

verificar los puertos abiertos

 # netstat -A inet6 -anp

Esto debe regresar una salida como la siguiente

Active Internet connections (servers and established)

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State	PID/Program name
tcp	0	0	:::13	:::*	LISTEN	6966/xinetd
udp	0	0	:::13	:::*		6966/xinetd
raw	0	0	:::58	:::*		7 -
raw	0	0	:::58	:::*		7 -
raw	0	0	:::58	:::*		7 -

tcp|udp = servicios habilitados de IPv6
raw = Kernel habilitado para IPv6

Tabla 3. Resultado de la ejecución del comando netstat

4.4.4. Cliente y servidor de Telnet

Información importante: no es recomendable usar telnet para conexiones remotas porque el password se transporta a través de la red como texto plano, se recomienda usar SSH[7] para esos casos.

URLs:

RPM fuente: http://www.netcore.fi/pekkas/linux/ipv6/

Usando SRPM (referencia: construir el RPM e instalar)

Reconstruir el binario

rpm --rebuild /path/a/SRPMS/telnet-version-release.src.rpm

Instalar el paquete

rpm -F|ihv /path/a/rpms/cpu/telnet-version-release.cpu.rpm

En el archivo /etc/xinetd.conf se debe quitar el comentario a la línea del telnetd, también se debe revisar es en que path se instala y revisar que sea el mismo path que tiene el xinetd.

4.4.5. Cliente del Finger

URLs

Fuente de los RPM para sistemas Red Hat: http://www.netcore.fi/pekkas/linux/ipv6/

[7]Secuity SHell, Shell seguro, conexiones encriptadas a través de una red.

Los SRPMS compilan en RHL 6.2 y 7.1 sin ningún problema.

Usando SRPMS: Construir el RPM e instalar

Reconstruir el binario

rpm --rebuild /path/to/SRPMS/finger-version-release.src.rpm

Instalar el paquete

rpm -F|ihv /path/to/rpms/cpu/finger-version-release.cpu.rpm

4.4.6. tcpdump y libpcap

Los paquetes tcpdump y libpcap se utilizan para la captura de paquetes, el primero es un programa de análisis de paquetes y el segundo es un biblioteca de desarrollo.

URL: www.tcpdump.org

fuentes en RPM: http://www.netcore.fi/pekkas/linux/ipv6/

usando los SRPM para instalar

rpm --rebuild /path/a/SRPMS/tcpdump-version-release.src.rpm
rpm --rebuild /path/a/SRPMS/libpcap-version-release.src.rpm

Instalar el paquete

rpm -F|ihv /path/to/RPMS/cpu/tcpdump-version-release.cpu.rpm
rpm -F|ihv /path/to/RPMS/cpu/libpcap-version-release.cpu.rpm

Usando los Fuentes para libpcap

1. Cambiarse y crear el directorio destino, y cambiarse dentro

   cd /usr/src; mkdir libpcap; cd libpcap

2. Desempaquetar el código

   tar xzf your-path/libpcap-version.tar.gz

3. Configurar

   ./configure --enable-ipv6

4. Compilar

   make clean; make

5. Instalar la biblioteca

   make install

La biblioteca se instala en /usr/local/lib/libpcap.a Para tcpdump se siguen los mismos pasos, solamente se cambia la referencia del nombre de los directorios, pero el procedimiento es el mismo.

A continuación se presenta una tabla donde se muestra un resumen de las aplicaciones instaladas

Aplicación	Que se hace
1. Kernel	• Reconfigurar las opciones del Kernel • Compilar • Instalar
2. xinetd	• Descargarlo de Internet • Descomprimir el paquete • Compilar • Instalar • Realizar pruebas para comprobar que el Kernel y los servicios estén listos para IPv6
3. net-tools 4. iputils	• Comprobar si ya manejan IPv6, si no descargar los paquetes de Internet • Descomprimir el paquete • Compilar • Instalar
5. telnet 6. finger	• Desinstalar los paquetes originales (si están instalados) • Descargar de Internet versiones que sean capaces de manejar IPv6 • Descomprimir el paquete • Compilar • Instalar • Probar el funcionamiento
7. tcpdump 8. libpcap	• Desinstalar los paquetes originales (si están instalados) • Descargar de Internet versiones que sean capaces de manejar IPv6 • Configurar con soporte para IPv6 • Compilar • Instalar

Tabla 4. Tabla que muestra un resumen de las aplicaciones

Capítulo 5. Aplicación

Introducción

En este capítulo se presenta el desarrollo de un pequeño sistema, un sniffer[1] a ciencia cierta, el cual permite verificar el funcionamiento de la red bajo IPv6, la idea es construirlo de tal manera que solamente capture paquetes IPv6.

El desarrollo se hizo por completo en el lenguaje c, se utilizó la biblioteca de desarrollo libpcap para la captura y recuperación de paquetes, el análisis se hace por medio de las estructuras definidas por el POSIX y la interfaz gráfica en Tcl/Tk.

5.1. Propuesta de desarrollo

Para el desarrollo del sniffer se proponen dos etapas, la primera de captura, la cual se encarga de establecer la comunicación con la interfaz de red y la captura de los datos, la segunda etapa consiste en el análisis de los mismos y el despliegue en pantalla. Lo anterior porque si se ejecuta en sistemas con un rendimiento bajo (poca memoria, procesador lento, o ambos) puede causar la caída del mismo, además de que para fines de estudio es mas fácil visualizar su funcionamiento en dos programas separados.

También cuenta con una interfaz gráfica desarrollada en Tcl/Tk, esta interfaz se encarga de hacer el llamado a los programas de captura y análisis de paquetes, además de presentar el resultado en pantalla.

5.2. Por qué libpcap

¿Por qué pcap? Porque es portable en diversos sistemas operativos.

Cuando se pretenden capturar paquetes a bajo nivel, es necesario usar las facilidades que brinde el sistema operativo para dicho fin. A continuación se presentan algunos ejemplos de métodos de captura de paquetes.

[1] Un programa o dispositivo que monitorea los datos que viajan a través de una red. Los sniffers pueden ser usados tanto para funciones legitimas como para robar información de una red. Sniffers no autorizados pueden ser extremadamente peligrosos para la seguridad en redes, porque son virtualmente imposibles de detectar y pueden ser insertados casi en cualquier lugar. En redes TCP/IP, donde hay sniffers, son con frecuencia llamados paquetes sniffer.

Método	Sistema Operativo
BPF (Berkeley Packet Filter)	Variantes de BSD
DLPI (Data Link Provider Interface)	Solaris, HP-UX, SCO, Openserver
NIT (Network Interface Tap)	SunOS
SNOOP	Irix
SNIT (Streams Network Inteface Tap)	
SOCKET_PACKET	Linux
LSF (Linux Socket Filter)	Linux

Figura 1. Métodos de diferentes sistemas operativos para la captura de paquetes

La forma más común de capturar paquetes, al menos en lo referente a Linux, es utilizando SOCKET_PACKET. El problema radica cuando se quiere portar a otros Unix, ya que se requiere cambiar la estrategia de captura de paquetes. La manera de resolver este problema es utilizar una biblioteca que aísle las particularidades de cada sistema operativo al momento de capturar paquetes. Esta función es desarrollada por libpcap, la biblioteca sobre la que se centra este desarrollo.

5.3. La biblioteca de desarrollo libpcap

A continuación se explican algunos de las funciones que comprenden a la biblioteca lipcap

La API (Aplication Program Interface) de libpcap [61]

A continuación se muestra la API que ofrece la biblioteca de captura de paquetes. No se detalla toda la API, sólo las funciones que tienen alguna injerencia dentro del programa.

5.3.1. Función *pcap_open_live()*

pcap_t* pcap_open_live(char *dev, int slen, int prm, int to_ms, char *ebuf)

donde:

- *dev* indica el dispositivo sobre el cual se hará la captura. El dispositivo se identificará por una cadena de caracteres, por ejemplo, “eth0”, “ppp”.

- *slen* indica el número máximo de bytes que se pueden capturar. Este valor dependerá de la tecnología de red que se este usando, ya que depende de los tamaños de la trama[2].

[2] El termino controlador de trama (frame) proviene de las comunicaciones seriales en las que el emisor estructura los datos al añadir caracteres especiales antes y después de los datos por transmitir.

- *prm* actúa como una bandera lógica, indicando si el dispositivo de captura se pondrá en modo promiscuo o no.

- *to_ms* es un valor entero que indica un timeout de la lectura en milisegundos.

- *ebuf* hace referencia a un buffer en donde en caso de error se guardará una cadena explicativa del problema surgido.

La función pcap_open_live() prepara las estructuras de datos necesarias para comenzar la captura de paquetes, devolviendo un descriptor que podría ser utilizado en las restantes operaciones de captura. En caso de error, devolverá NULL.

5.3.2. Función *pcap_open_offline()*

pcap_t* pcap_open_offline(char *fname, char *ebuf)

La función *pcap_open_offline()* permite abrir un archivo en lugar de un dispositivo de red. Esta función devolverá un descriptor o NULL en caso de error. La intención de esta función es recuperar los datos de una captura anterior una vez que estén almacenados en un archivo, donde:

- *fname* es el nombre del archivo que se abre para leer los paquetes

- *ebuf* es un buffer donde guardar en caso de error un mensaje explicativo

5.3.3. Función *pcap_dump_open()*

pcap_dumper_t* pcap_dump_open(pcap_t *p, char *fname)

La función *pcap_dump_open()* abre un archivo para volcar en él los paquetes capturados. Este archivo podrá ser abierto con la función *pcap_open_offline* para leer su contenido. En caso de error se devolverá NULL.

- *p* es un descriptor devuelto por las funciones *pcap_open_live()* o *pcap_open_offline()*.

- *fname* es el nombre del archivo que se quiere abrir para escritura.

5.3.4. Función *pcap_lookupdev()*

char* pcap_lookupdev(char *errbuf)

La función *pcap_lookupdev ()* devuelve una cadena de caracteres que identifica un dispositivo de sistema válido para realizar la captura de paquetes. La cadena devuelta por esta función puede ser usada como parámetro *dev* en la función *pcap-open_live()*. Si hay algún error devolverá NULL.

- *errbuf* es un buffer donde, en caso de error, la función devuelve una cadena explicativa del error.

5.3.5. Función *pcap_lookupnet()*

pcap_lookupnet (char *dev, bpf_u_int32 *netp, bpf_u_int32 *maskp, char ebuf)

La función *pcap_lookupnet()* obtiene la red y máscara de red asociadas con un dispositivo de captura. En caso de error la función devuelve –1.

- *dev* es una cadena de texto representativa del dispositivo.
- *netp* recibirá el valor de la red.
- *maskp* recibirá el valor de la mascara de red.
- *ebuf* buffer en el que se almacena el mensaje de error.

esta función solo obtiene valores validos para IPv4.

5.3.6. Función *pcap_dispatch()*

int pcap_dispatch(pcap_t p, int cnt, pcap_handler callback, u_char *user)

La función *pcap_dispatch()* comienza la captura de paquetes. Devuelve el número de paquetes recibidos, 0 en caso de alcanzar un EOF ó –1 en caso de error.

- *p* es un descriptor obtenido mediante *pcap_open_live()* o *pcap_open_offline()*
- *cnt* indica el número máximo de paquetes que se procesarán antes de que la función termine. Un valor de –1 procesa todos los paquetes recibidos en un buffer. Un valor de 0 procesa todos los paquetes hasta que ocurre un error o expira el *timeout* de lectura establecido.
- *callback* hace referencia a una rutina que será llamada para analizar el paquete recibido. Esta rutina tiene la sintaxis siguiente:

  void callback(u_char *ud, pcap_pkthdr *pkthdr, u_char *pd)

- *ud* hace referencia a datos de usuario que serán pasados desde *pcap_dispatch()* a través del parámetro *user* de esa función.
- *pkthdr* es una cabecera adicional incorporada por la biblioteca, precede a los datos y encabezados propios de los protocolos de red. Su formato es el siguiente:

```
struct pcap_pkthdr {
  struct timeval ts;          /* etiqueta de tiempo */
  bpf_u_int32 caplen;         /* longitud de la porción presente */
  bpf_u_int32 len;            /* longitud de este paquete */
};
```

- *pd* es un puntero a través del cual estarán accesibles los datos del paquete capturado

- *user* es un puntero a los datos del usuario

5.3.7. Función *pcap_loop()*

```
int pcap_loop(pcap_t *p, int cnt, pcap_handler callback, u_char *user)
```

La función *pcap_loop()* es similar a *pcap_dispatch()*, esta función procesará tantos paquetes como se le indiquen en el parámetro *cnt*, o terminará antes en caso de error, pero nunca terminará por *timeout* de la lectura. Un valor negativo en el parámetro *cnt* provocará un ciclo infinito, procesando todos los paquetes recibidos y parando únicamente en caso de error. El resto de los parámetros tienen el mismo significado que la función *pcap_dispatch()*.

5.3.8. Función *pcap_dump()*

```
void pcap_dump(u_char *user, struct pcap_handler *h, u_char *sp)
```

Guarda un paquete en un archivo abierto previamente con la función *pcap_dump_open()*. Los parámetros de esta función son los mismos que lo de la rutina *callback* indicada en *pcap_dispacth()* y *pcap_loop()*.

5.3.9. Función *pcap_compile()*

```
int pcap_compile(pcap_t, struct bpf_program *fp, char *str, int optimize,
                 bpf_u_int32 netmask)
```

La función *pcap_compile()* toma una cadena de texto y la interpreta para crear un filtro sobre el cual basarse a la hora de elegir que paquetes toma para su análisis.

La cadena de texto se indica en el parámetro *str*, esta cadena de texto se interpreta como un filtro que se guarda en la estructura indicada por el parámetro *fp*. El filtro almacenado en *fp* podrá optimizarse o no, dependiendo del valor del parámetro *optimize*. Por último, el parámetro *netmask* toma la mascara de red asociada al dispositivo de captura, ese dato puede obtenerse mediante la función *pcap_lookupnet()*.

5.3.10. Función *pcap_setfilter()*

int pcap_setfilter(pcap_t *p, struct bpf_program *fp)

La función *pcap_setfilter()* activa el filtro indicado en el parámetro *fp*. Este filtro habrá sido creado con anteriormente mediante la función *pcap_compile()*. Si el filtro se ha establecido correctamente, la función *pcap_setfilter()* devuelve 0; en caso de error devuelve –1.

5.3.11. Función *pcap_next()*

u_char* pcap_next(pcap_t *p, struct pcap_pkthdr *h)

La función *pcap_next()* devuelve un puntero al siguiente paquete que se haya capturado y se tenga almacenado en el *buffer* de entrada.

5.3.12. Función *pcap_perror*

char *pcap_perror(pcap_t *p, char *prefix)

La función *pcap_perror()* es similar a la función *perror().* Muestra por *stderr* una cadena explicativa del último error ocurrido con alguna función de la biblioteca pcap. Es mensaje es precedido por la cadena *prefix*.

5.3.13. Función *pcap_close()*

void pcap_close(pcap_t *p)

La función *pcap_close()* libera todos los recursos asociados con el dispositivo de captura referenciado por el apuntador *p*.

5.3.14. Función *pcap_dump_close()*

void pcap_dump_close(pcap_dumper_t *p)

Esta función cierra el archivo de captura asociado a *p*.

5.4. Estructuras definidas para encabezados de IPv6

Uno de los dos problemas a resolver con el sniffer ya ha sido resuelto con la API de libpcap, la captura y recuperación de paquetes. Para el análisis de paquetes la historia es otra, dentro del sistema existen estructuras de datos definidas para cada tipo de paquetes, así, existen estructuras de datos para el encabezado Ethernet, el encabezado IP, etc. A continuación se presentan las estructuras de datos para los tipos de paquetes que se capturan y que de alguna manera se analizan.

5.4.1. Encabezado Ethernet.

El encabezado Ethernet se encuentra definido, para Red Hat Linux 7.2, en /usr/include/net/ethernet.h, para diferentes versiones/distribuciones puede variar, y tiene la siguiente estructura

```
/* 10Mb/s ethernet header */
struct ether_header
{
  u_int8_t  ether_dhost[ETH_ALEN];      /* destination eth addr */
  u_int8_t  ether_shost[ETH_ALEN];      /* source ether addr    */
  u_int16_t ether_type;                 /* packet type ID field */
} __attribute__ ((__packed__));
```

En la aplicación se utiliza este encabezado para conocer las direcciones MAC origen y destino de los paquetes que están circulando por la red. Si se capturan paquetes en una red corriendo IPv4 y se analiza el encabezado Ethernet, se puede ver que para este encabezado no existe ningún cambio comparado con aquellos que transportan paquetes IPv6.

5.4.2. Encabezado IPv6

Analizar este encabezado es de gran importancia, ya que esta es la estructura de datos que representa en bits y bytes lo escrito en el capitulo 3 donde se presenta la estructura y los encabezados de extensión que lo comprenden. Este es el punto fuerte del sniffer, ya que interesa saber que tipo de paquetes fluyen a través de la red, que encabezados de extensión lo siguen, incluso la información que transporta.

El encabezado IPv6 se encuentra declarado, para sistemas corriendo Red Hat Linux 7.2 en /usr/include/netinet/ip6.h, para diferentes versiones/distribuciones puede variar

```
struct ip6_hdr
 {
  union
   {
    struct ip6_hdrctl
     {
      uint32_t ip6_un1_flow;   /* 4 bits version, 8 bits TC,
                                  20 bits flow-ID */
      uint16_t ip6_un1_plen;   /* payload length */
      uint8_t  ip6_un1_nxt;    /* next header */
      uint8_t  ip6_un1_hlim;   /* hop limit */
     } ip6_un1;
    uint8_t ip6_un2_vfc;       /* 4 bits version, top 4 bits tclass */
   } ip6_ctlun;
  struct in6_addr ip6_src;     /* source address */
  struct in6_addr ip6_dst;     /* destination address */
 };

#define ip6_vfc   ip6_ctlun.ip6_un2_vfc
#define ip6_flow  ip6_ctlun.ip6_un1.ip6_un1_flow
#define ip6_plen  ip6_ctlun.ip6_un1.ip6_un1_plen
#define ip6_nxt   ip6_ctlun.ip6_un1.ip6_un1_nxt
#define ip6_hlim  ip6_ctlun.ip6_un1.ip6_un1_hlim
#define ip6_hops  ip6_ctlun.ip6_un1.ip6_un1_hlim
```

Los valores de los encabezados de extensión, por su valor numérico se encuentran definidos en /usr/include/net/in.h

```
   IPPROTO_HOPOPTS = 0,   /* IPv6 Hop-by-Hop options.  */
#define IPPROTO_HOPOPTS         IPPROTO_HOPOPTS
   IPPROTO_IPV6 = 41,     /* IPv6 header.  */
#define IPPROTO_IPV6            IPPROTO_IPV6
   IPPROTO_ROUTING = 43,  /* IPv6 routing header.  */
#define IPPROTO_ROUTING         IPPROTO_ROUTING
   IPPROTO_FRAGMENT = 44, /* IPv6 fragmentation header.  */
#define IPPROTO_FRAGMENT        IPPROTO_FRAGMENT
   IPPROTO_ICMPV6 = 58,   /* ICMPv6.  */
#define IPPROTO_ICMPV6          IPPROTO_ICMPV6
   IPPROTO_NONE = 59,     /* IPv6 no next header.  */
#define IPPROTO_NONE            IPPROTO_NONE
   IPPROTO_DSTOPTS = 60,  /* IPv6 destination options.  */
#define IPPROTO_DSTOPTS         IPPROTO_DSTOPTS
```

Los encabezados de extensión forman parte de la definición del encabezado IPv6, estos se encuentran definidos en /usr/include/netinet/ip6.h.

```
/* Hop-by-Hop options header.  */
struct ip6_hbh
  {
    uint8_t  ip6h_nxt;        /* next header.  */
    uint8_t  ip6h_len;        /* length in units of 8 octets.  */
    /* followed by options */
  };

/* Destination options header */
struct ip6_dest
  {
    uint8_t  ip6d_nxt;        /* next header */
    uint8_t  ip6d_len;        /* length in units of 8 octets */
    /* followed by options */
  };

/* Routing header */
struct ip6_rthdr
  {
    uint8_t  ip6r_nxt;        /* next header */
    uint8_t  ip6r_len;        /* length in units of 8 octets */
    uint8_t  ip6r_type;       /* routing type */
    uint8_t  ip6r_segleft;    /* segments left */
    /* followed by routing type specific data */
  };

/* Type 0 Routing header */
struct ip6_rthdr0
  {
    uint8_t  ip6r0_nxt;       /* next header */
    uint8_t  ip6r0_len;       /* length in units of 8 octets */
    uint8_t  ip6r0_type;      /* always zero */
    uint8_t  ip6r0_segleft;   /* segments left */
    uint8_t  ip6r0_reserved;  /* reserved field */
    uint8_t  ip6r0_slmap[3];  /* strict/loose bit map */
    struct in6_addr  ip6r0_addr[1];  /* up to 23 addresses */
  };

/* Fragment header */
struct ip6_frag
  {
    uint8_t   ip6f_nxt;       /* next header */
    uint8_t   ip6f_reserved;  /* reserved field */
    uint16_t  ip6f_offlg;     /* offset, reserved, and flag */
    uint32_t  ip6f_ident;     /* identification */
  };
```

5.4.3. Encabezado ICMPv6

Hasta antes de este encabezado, se ha cumplido el propósito de saber que tipo de información fluye en la red, ahora es necesario mostrar como se accede a los encabezados de extensión, en este caso y como muestra se analiza ICMPv6. Cuando se detecta en el campo de próximo encabezado un ICMPv6, se analiza de manera básica. Este encabezado se encuentra definido en /usr/include/netinet/icmp6.h

```
struct icmp6_hdr
 {
   uint8_t     icmp6_type;   /* type field */
   uint8_t     icmp6_code;   /* code field */
   uint16_t    icmp6_cksum;  /* checksum field */
   union
    {
      uint32_t  icmp6_un_data32[1]; /* type-specific field */
      uint16_t  icmp6_un_data16[2]; /* type-specific field */
      uint8_t   icmp6_un_data8[4];  /* type-specific field */
    } icmp6_dataun;
 };
```

5.5. Algoritmo del programa de captura.

1. Iniciar revisando que el número de parámetros de la función main sean correctos.

2. Buscar el dispositivo de red con la función pcap_lookupnet.

3. Abrir la comunicación con la función pcap_open_live.

4. Crear el archivo donde se van a guardar los datos.

5. Colocar los filtros para capturar paquetes de IPv6 con la función pcap_compile y pcap_set_filter.

6. Realizar con la función pcap_loop un ciclo para que con la función pcap_dump se capturen el número de paquetes indicados en los parámetros.

7. Se liberan los recursos usados durante la captura con la función pcap_close.

8. Fin del programa.

5.6. Diagrama a bloques de la aplicación de captura.

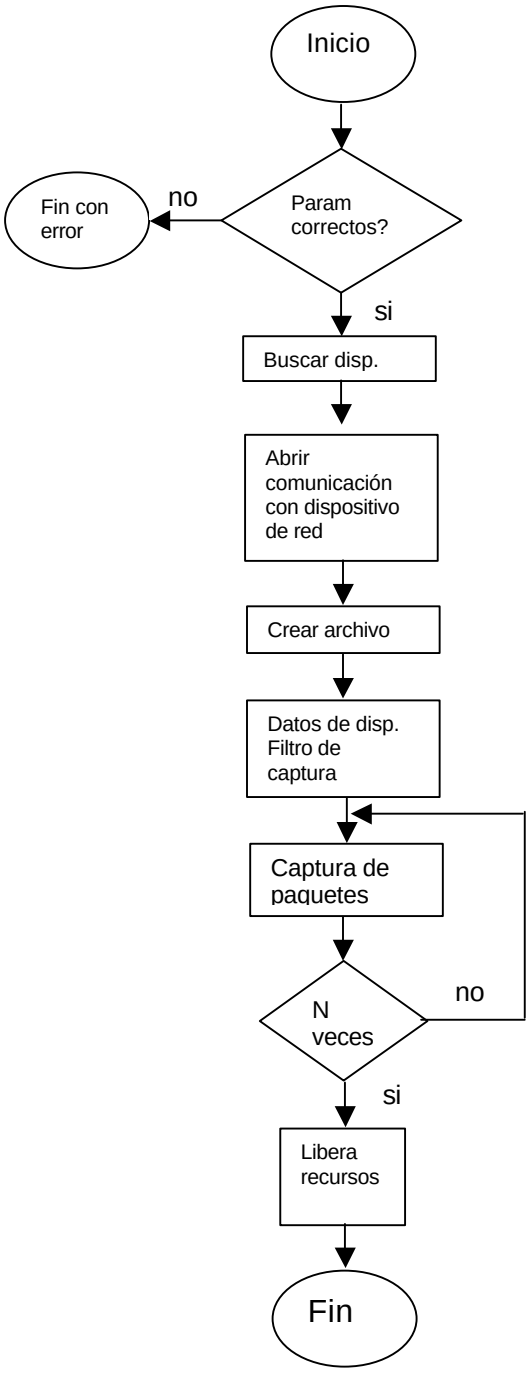

Figura 2. Diagrama de flujo propuesto para el módulo de análisis de paquetes

5.7. Algoritmo de la aplicación de análisis

1. Iniciar revisando que el número de parámetros de la función main sean correctos.

2. Abrir el archivo indicado en el parámetro main, puede ser el mismo que el del programa de captura o algún otro archivo que se tenga guardado.

3. Iniciar un ciclo de recuperación de paquetes del archivo

    a. Extraer un paquete con la función pcap_next.

    b. Igualar el paquete a la estructura Ethernet

    c. Obtener las direcciones MAC origen y destino del paquete

    d. Igualar el paquete a la estructura IPv6

    e. Obtener la información de los campos que contiene, por medio de una función se imprime el nombre del tipo de próximo encabezado.

    f. Si el próximo encabezado es un ICMPv6 se analiza y se obtienen el tipo y el código de mensaje.

4. Se liberan recursos

5. Fin de programa.

5.8. Diagrama a bloques de la aplicación de análisis.

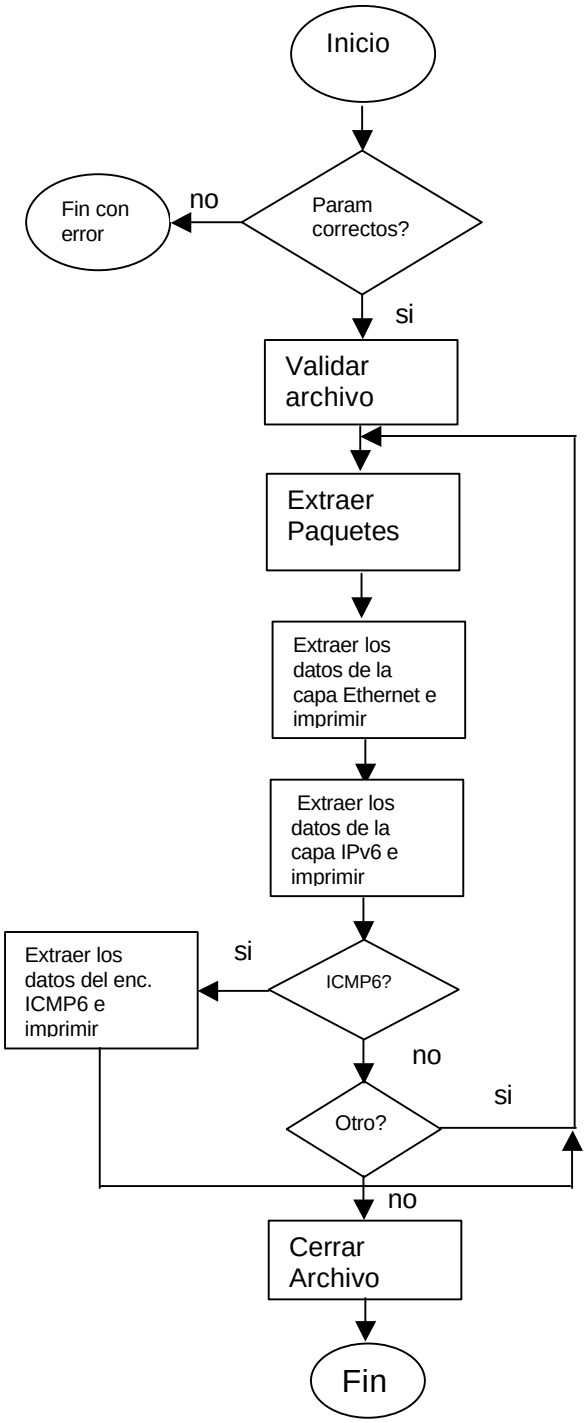

Figura 3. Diagrama de flujo propuesto para el módulo de análisis de paquetes

5.9. Resultados

El tipo de implementación de las aplicaciones de captura y análisis permite que se ejecuten en modo texto o en modo gráfico, si es usado en un servidor o en una estación de trabajo, a continuación se presentan las pantallas mostrando la ejecución del programa, la primera, que se muestra a continuación es la pantalla de captura, esta captura en particular pone el programa a “escuchar” 10 paquetes, los guarda en el directorio */home/hiram/test* y el filtro es para paquetes IPv6.

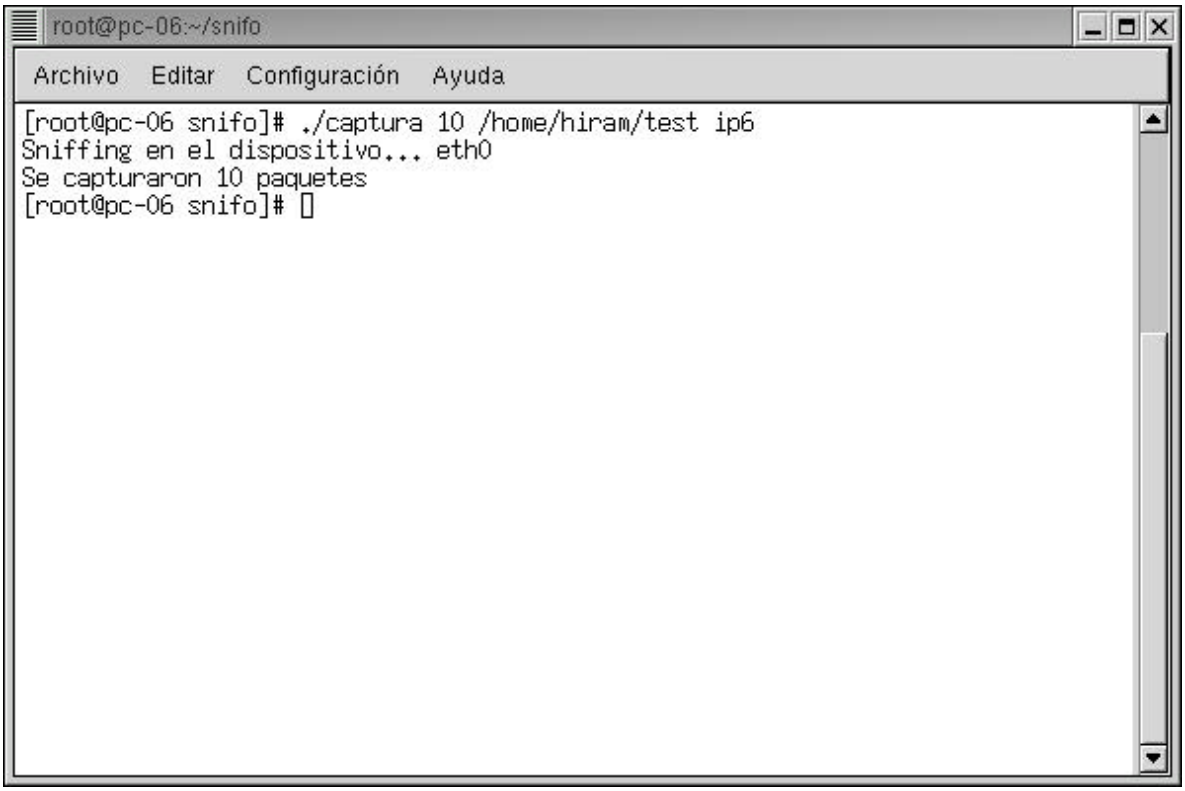

Figura 4. Pantalla de la ejecución del programa de captura

La siguiente pantalla muestra la salida de la ejecución del programa de análisis, en el se muestra el numero de paquetes, en este caso solamente se analizó un archivo con dos paquetes. En el se muestran los resultados del análisis de los encabezados Ethernet e IPv6.

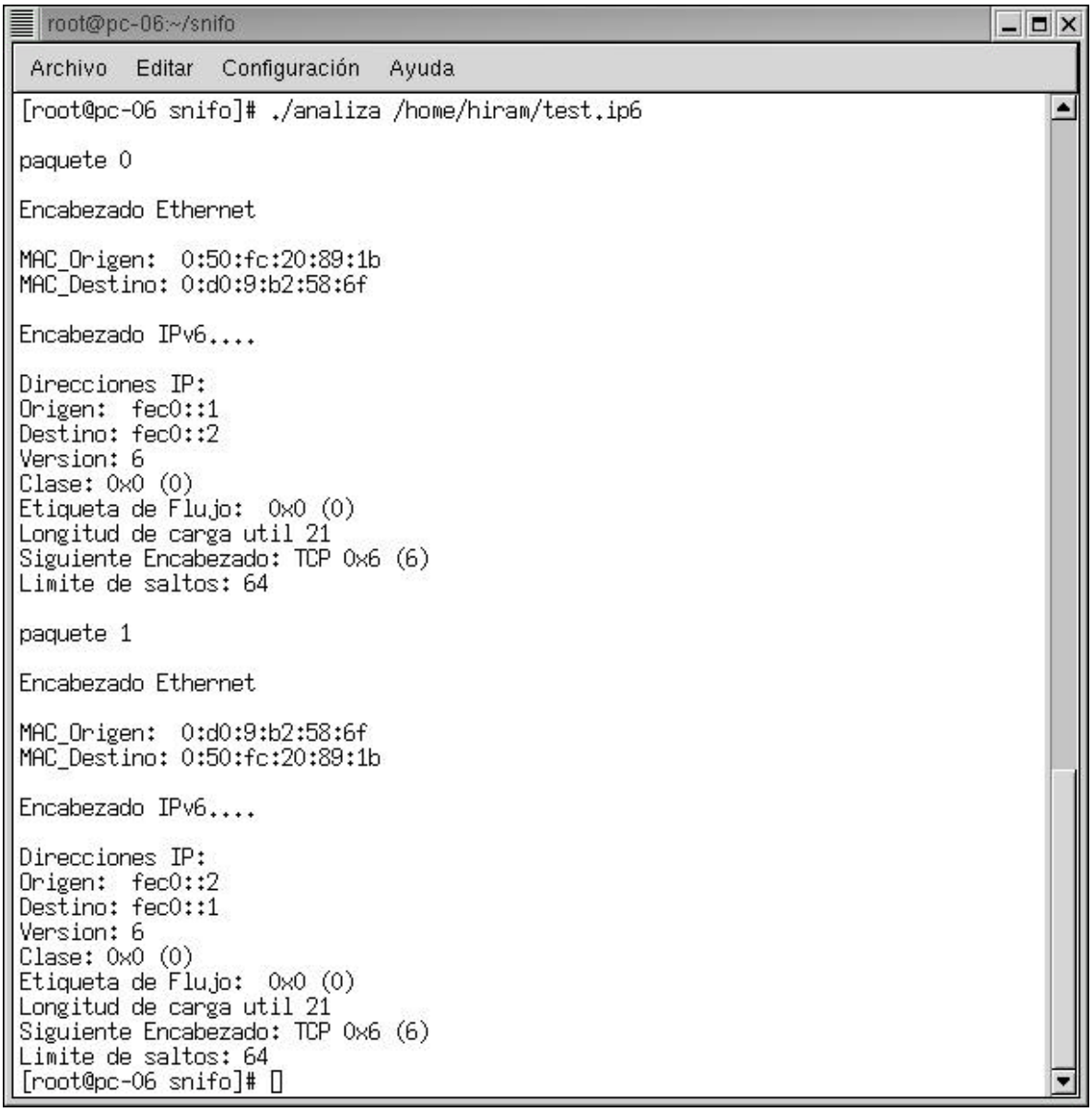

Figura 5. Pantalla de la ejecución del programa de análisis

La siguiente pantalla muestra la ejecución de la interfaz gráfica, en ella se aprecian campos que delimitan el lugar donde se presenta información importante, como el número de paquete, las direcciones MAC y las direcciones IPv6.

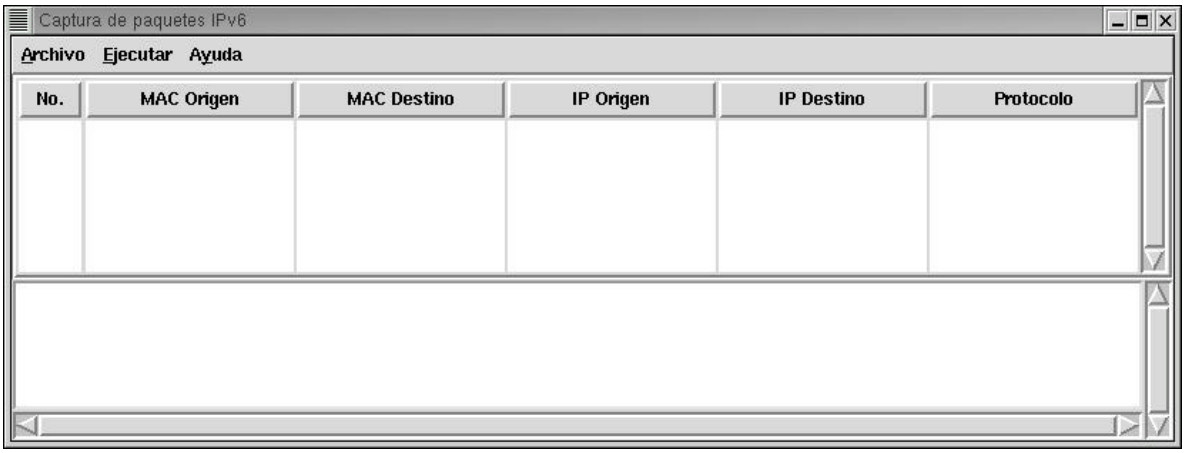

Figura 6. Pantalla de la interfaz gráfica del programa

La siguiente pantalla muestra la presentación del programa de análisis de paquetes.

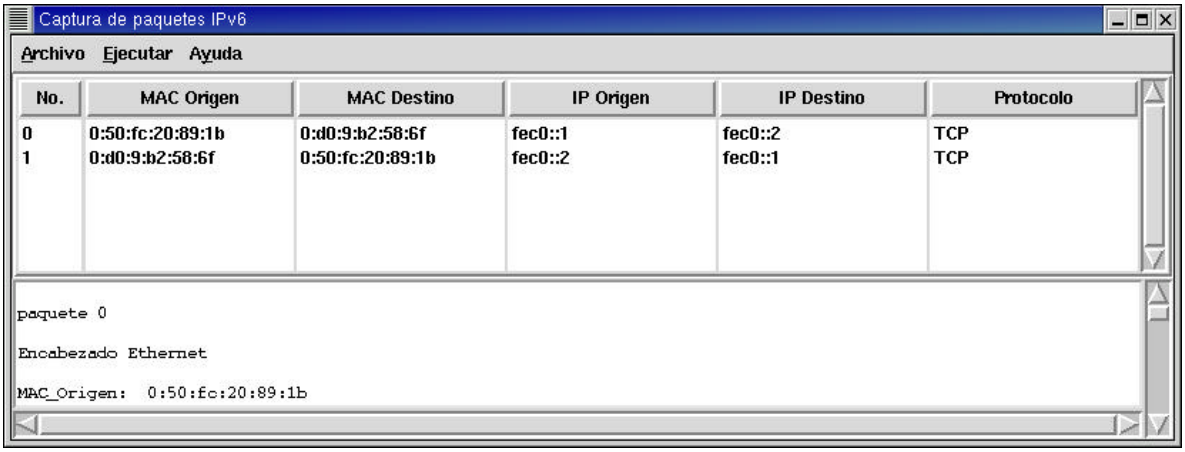

Figura 7. Pantalla donde se muestra el análisis de paquetes en la interfaz gráfica

Así, en base a las imágenes de los programas, se verifica que el tráfico que fluye a través de la red, transporta paquetes de IPv6.

Capítulo 6. Perspectivas de IPv6

Introducción

La importancia del capítulo se hace presente al momento de comprender la necesidad de la evolución, de preparar gente que sepa entender el cambio, es decir, no se trata simplemente de un ardid publicitario, es un problema verdadero y que tarde o temprano nos alcanzará.

Se presentan casos de organizaciones comprometidas con la investigación y el desarrollo, Institutos y Universidades en el país, e incluso en el estado, que llevan la delantera en lo referente a configuración, pruebas y puesta en marcha de túneles hacia el back-bone de pruebas de IPv6.

También se presentan algunos de los proyectos en otras partes del mundo, en especial en América Latina.

6.1. IPv6 en América Latina

IPv6, como toda nueva tecnología, siempre es aplicada más tarde que temprano en países del llamado “tercer mundo” o subdesarrollado, el caso de México no es la excepción. Se presentan dos casos, el de la UNAM, por ser el primer nodo en el país que maneja IPv6, y el caso del Instituto Tecnológico de Oaxaca, por ser el primer nodo en el estado.

6.1.1. IPv6 en la UNAM.

La UNAM inició investigaciones en la materia desde el mes de diciembre de 1998, fecha en la que se constituye el proyecto IPv6 y durante el segundo semestre del año 1999 es notable el liderazgo de la UNAM en el ámbito nacional. Dentro del Proyecto IPv6 de la UNAM se estableció un amplio programa de pruebas y trabajos con temas como: implementaciones, stacks IPv4/IPv6, túneles, software de conexión, aplicaciones multimedia, servidores para Web y DNS, autoconfiguración, calidad de servicio, IPv6 sobre ATM, conexión con redes internacionales de IPv6 (6Bone, 6REN), IPv6 en Internet2, etc.

Dentro de las primeras pruebas realizadas, destaca la conexión a 6Bone, la cual es una red mundial experimental utilizada para probar los conceptos y la puesta en operación de IPv6. Actualmente participan en 6Bone en el ámbito

mundial 47 países, entre ellos México, donde la UNAM fue el primer nodo en el país, registrándose en junio de 1999.

Posteriormente en septiembre de 1999 la UNAM fue aceptada como uno de los 68 nodos de Backbone que a la fecha operan en 6Bone, obteniendo un rango de direcciones tipo pTLA[1]: 3ffe:8070::/28. Cabe destacar que con este hecho la UNAM es el primer nodo, y hasta el momento el único, de este tipo en México, y el tercero en Latinoamérica. Adicionalmente, la UNAM puede delegar direcciones y configurar túneles a instituciones en México y en el mundo interesadas en realizar pruebas con IPv6.

Para contar con una red de pruebas en una primera etapa, y posteriormente con una red de producción, se instaló la Red IPv6 de la UNAM, la primera red IPv6 instalada en México y que inició operaciones en agosto de 1999. Esta red cuenta con varios túneles hacia otros nodos de Backbone de 6Bone: SPRINT, FIBERTEL, MERIT, BAY NETWORKS, JANET e ISI-LAP, y hacia los hosts que tiene la UNAM corriendo con sistemas operativos como Win NT4, Win 2000, Solaris y Linux.

Contacto: Físico. Cesar Olvera cesar@redes.unam.mx

6.1.2. IPv6 en el ITO

El Instituto Tecnológico de Oaxaca cuenta con un túnel hacia la UNAM para realizar pruebas con IPv6, tiene los siguientes objetivos:

- Colaborar con la Investigación en IPv6 a nivel nacional
- Investigar, conocer y entender el protocolo para su investigación
- Realizar un túnel hacia la UNAM para realizar pruebas

Los servicios que se plantea brindar a los usuarios de su red son

- DNS – servidor de nombres
- WEB -Servidor de http
- MAIL – servidor de correo electrónico

El servidor habilitado para manejar el túnel es un SUN Ultra Sparc Ultra1 con un disco de 4Gb a 167 Mhz.

Contacto: L.I. Cristina Espinosa Martínez cris@itoaxaca.edu.mx

[1] pseudo Top-Level Aggregation Identifier, Identificador pTLA

6.2. IPv6 en otras Instituciones en México

A continuación se presenta una lista según UK IPv6 Resource Centre[50], Lancaster University Computing Department, de los sitios que manejan IPv6 en México:

- ASCICESE: CICESE academic member of Red-CUDI (Internet 2 Mexico)
- CIC-IPN: Centro de Investigación en Computación, 6Bone Site, Ciudad de México
- CICESE: CICESE academic member of Red-CUDI (Internet 2 México)
- CUDI: Corporación Universitaria para el Desarrollo de Internet (CUDI), IPv6 and Internet2 Testbed
- DGSCA: Red para prueba de aplicaciones en IPv6, IPv6 Aplication testing Network
- FI-UNAM: Sitio de la Facultad de Ingenieria de la UNAM, IPv6 Testbed
- ITAM: Instituto Tecnológico Autónomo de México, 6Bone site, Ciudad de México
- ITESM: ITESM Campus Monterrey Departamento de Telecomunicaciones y Redes Monterrey, Nuevo León
- ITESM-CCM: IPv6 ITESM CCM
- ITESM-RUV: Tec de Monterrey Virtual University
- ITESM-RZN: Tec de Monterrey Rectoría Zona Norte
- NIC-MX: ccTLD MX Registry
- UDG: Universidad de Guadalajara, Coordinación de Telecomunicaciones y Redes NOC UDG
- ULSA: ULSA academic member of Red-CUDI (Internet 2 México)
- UNAM: Universidad Nacional Autónoma de México, 6Bone pTLA Site, Ciudad de México

6.3. En otros sitios de América Latina

A continuación se presenta una lista de sitios que habilitados para manejar IPv6 en otros sitios de América Latina según UK IPv6 Resource Centre, Lancaster University Computing Department.

ARGENTINA

- AWORLD: Atomic World; Avellaneda_BA, AR
- BARRAHOME: Barra Home The Spanish PHP Group
- BESOLOCO: Beso Loco, qué beso loco.
- BOMBI-BOMBI: Bombi-Bombi, Buenos Aires, Argentina.
- CENTAURI-AR: Centauri, La Plata, AR
- COMPENDIUM-AR: Compendium, Buenos Aires, AR
- FIBERTEL: Fibertel TCI Argentina
- GEMINIS-AR: Geminis, Buenos Aires, Argentina.
- GEMINIS6: Geminis IPV6 test site
- ORBISTEL: Orbistel, Córdoba, AR
- UTN-FRLP: Universidad Tecnológica Nacional - Fac. Reg. La Plata, La Plata / Republica Argentina

BRASIL

- CEFET-BA: CEFET-BA - CENTRO FEDERAL DE EDUCACAO TECNOLOGICA DA BAHIA, FEDERAL CENTER OF TECHNOLOGICAL EDUCATION OF BAHIA
- IAE-SP: Instituto Adventista de Ensino - Campus I
- PARAISONET: Paraisonet Ltda. (Pegasus Network)
- POP-MG: POP Minas Gerais
- POP-RN: PoP Rio Grande do Norte
- REDEPEGASUS: Rede Pegasus, Pegasus Network Brazil
- RNP: RNP – Rede Nacional de Pesquisa, Brazilian National Research Network
- SURRIEL: Rik Van Riel’s home
- UAINET: Uainet Guaxupe Ltda. (Pegasus Network)
- UCB-BR: UCB-BR IPv6 Site
- UNICAMP: Universidade Estadual de Campinas
- UNINCOR: Universidade Vale do Rio Verde – Unincor (Pegasus Network)

CHILE

- INF-UTFSM: Universidad Tecnica Federico Santa, Maria - Valparaiso - Chile
- UACH: Universidad Austral de Chile, Instituto de Informatica
- UACH-IPV6: Universidad Austral de Chile Instituto de Informatica pNLA delegation for the 6bone

CUBA

- CAONAO: Centro de Gestión Tecnológica

URUGUAY

- RAU: RAU - Red Académica Uruguaya, SeCIU - Universidad de la Republica, Uruguayan research network

6.4. IPv6 en el mundo

Alrededor del mundo existen diferentes equipos de trabajo que realizan un esfuerzo en el desarrollo, implementación, migración, pruebas, etc sobre IPv6, a continuación se presentan sólo algunos de estos.

6.4.1. Proyecto KAME

El proyecto KAME es un proyecto en común para crear un sólido conjunto simple de software, especialmente enfocado sobre IPv6/IPsec. Investigadores con gran talento de grandes compañías Japonesas están unidos en este proyecto. Este esfuerzo conjunto evitará el innecesario desarrollo duplicado en la misma área, proporciona gran calidad y paquetes con un avanzado desempeño. La ambición del proyecto KAME es proporcionar referencias gratis para implementaciones de

- IPv6
- IPsec (para IPv4 e IPv6)
- Interconexión avanzada de redes tales como encolado avanzado de paqutes, ATM, movilidad y lo que sea interesante

en variantes de BSD.

Actualmente muchas variantes de BSD son desarrolladas incluyendo FreeBSD, NetBSD, OpenBSD y BSDI como productos comerciales. Ellos están desarrollando/mejorando el código de red (el árbol sys/netinet) separadamente.

En los días de IPv4 existía una buena referencia de código, tal como UCB Network Releases. Pero cuando se considera usar IPv6, hay muchas opciones todavía. El problema es, que aún si los proyectos BSD escogen una pila de IPv6 para agruparse en torno a ella, el código será mantenido por cada proyecto separadamente y ese código será mantenido por cada proyecto separadamente y muy probablemente será un poco diferente en cada árbol de proyecto.

Así, el proyecto KAME se formó para implementar y mantener el mejor código disponible para IPv4/IPv6/IPSec/etc, los cuales serán la base para la interconexión avanzada de redes en el siglo 21.

El URL del proyecto KAME es el siguiente: http://www.kame.net

6.4.2. Proyecto WIDE

El proyecto WIDE es otro proyecto de Japón, la lista de propósitos del proyecto se muestra a continuación.

- Implementar software IPv6
- Desarrollar ambientes IPv6
- Desarrollar mecanismos de transición de IPv4 a IPv6
- Establecer la tecnología y conocimientos para la administración de redes IPv6

Este proyecto es también para variantes de BDS, el URL del proyecto es el siguiente: http://www.v6.wide.ad.jp/

6.4.3. IPv6 Forum

El IPv6 Forum es un consorcio mundial de proveedores lideres de Internet, formado por redes de investigación y educación, con la clara misión de promocionar IPv6, mejorando dramáticamente el mercado y la conciencia de los usuarios de IPv6, creando una Internet de próxima generación segura y de alta calidad y permitiendo el acceso mundial y equitativo al conocimiento y la tecnología.

El fin de IPv6 FORUM es

- Establecer foro abierto internacional de experticia de IPv6.
- Compartir conocimiento y experiencia de IPv6 entre los miembros.
- Promover nuevas aplicaciones basadas en IPv6 y soluciones globales.

- Promocionar implementaciones interoperables de estándares de IPv6.
- Cooperar para cumplir la calidad de servicio extremo a extremo.
- Resolver cuestiones que creen barreras al desarrollo de IPv6.

Este es un sitio donde sólo miembros registrados pueden acceder a toda la información disponible, el registro por organización es de $2500 USD, la URL de este sitio es http://www.ipv6forum.org

6.4.4. Proyecto TAHI

El proyecto TAHI es el esfuerzo conjunto formado con el objetivo de desarrollar y proporcionar la tecnología de verificación para IPv6. El proceso de crecimiento de IPv4 fue la historia de toparse con muchos tipos de obstáculos y la superación de dichos obstáculos. Sin embargo, una vez que la posición como la infraestructura fueron establecidas, no esta permitido repetir la misma historia. Esta es la razón por la que la tecnología de verificación es esencial para el desarrollo de IPv6.

- El proyecto TAHI investiga y desarrolla pruebas de conformidad e interoperabilidad para IPv6.
- Trabaja cercanamente con el proyecto KAME y el proyecto USAGI, ayudándolos en el lado de la calidad, ofreciendo la tecnología de la verificación y mejorando la eficiencia del desarrollo.
- Se abren los resultados y frutos del proyecto al público gratis. Cualquier desarrollador que le concierna IPv6 puede utilizar libremente los resultados y frutos del proyecto TAHI.

  El URL del proyecto TAHI es el siguiente: http://www.tahi.org/

6.4.5. Proyecto USAGI

El proyecto USAGI (UniverSAl playGround for IPv6) es un agresivo proyecto de desarrollo para IPv6, principalmente para sistemas Linux.

Actualmente se encuentran desarrollando una implementación en el árbol de código fuente del Kernel de Linux. Una vez que se ha habilitado el Kernel para IPv6 y empezarlo a usar, se debe estar conciente de que la implementación tiene algunos problemas, ya que una implementación puede ser muy vieja o no estar bien probada, es decir, tiene muchos bugs y funciones no implementadas.

La meta del proyecto es mejorar el ambiente IPv6 en Linux y desarrollar Internet sobre IPv6 en el mundo. Se ha iniciado el trabajo con el Kernel, bibliotecas y aplicaciones agresivamente. Los productos y resultados se proporcionarán libremente para Linux y la comunidad de IPv6.

Las metas son tratar de mejorar:

- La pila de protocolos en el Kernel de Linux
- Las API's de IPv6 en la biblioteca glibc
- Aplicaciones con capacidades de IPv6

El URL de este proyecto esta en http://www.linux-ipv6.org/

Continuar con la lista de instituciones y proyectos que trabajan con IPv6, por ejemplo el IETF, SUN, Microsoft, etc, se puede convertir en una tarea interminable, en la siguiente liga puede encontrarse una lista mas completa de sitios de empresas e instituciones que trabajan con ipv6: http://www.ipv6.org/v6-www.html.

6.5. Trabajos a futuro

El trabajo a futuro que se puede desarrollar con esta tesis abarca una gran variedad de aspectos relativos a actividades empresariales o de investigación. A continuación se enumeran algunos puntos.

Como actividades empresariales

- Ya que la migración de todos los sistemas implica una gran cantidad de trabajo, es factible conformar una empresa especializada en consultoría y apoyo a empresas para la migración de servidores y aplicaciones a IPv6.
- Servicios varios de configuración, actualización y migración de servidores.

Como actividades de investigación

- Investigación sobre el nuevo protocolo, tales como audio y video, seguridad, multihoming, movilidad, etc.
- Bases para la conexión de Internet 2.
- Cooperación con otras instituciones que realicen investigación en las mismas áreas.

Conclusiones

- El desarrollo de aplicaciones de código abierto puede ayudar a que el conocimiento se transmita mas fácilmente.

- La falta de planeación de IPv4 hizo que un buen diseño de un protocolo terminara siendo obsoleto, no por su funcionalidad, si no por su uso y por la delegación de direcciones

- La planeación de la evolución del protocolo IP se hace en base a las fortalezas del diseño anterior y se le agregan campos y funcionalidad que se espera sean suficientes para soportar las aplicaciones presentes y futuras, se eliminan campos e información redundante y se hace mas eficiente al alinearlo para procesadores de 64 bits.

- La creación de grupos de trabajo para el desarrollo y seguimiento de los proyectos es necesario.

- Es necesario, aprovechar ahora que existe la oportunidad y la invitación para que instituciones, universidades y empresas realicen pruebas para comprobar la efectividad del protocolo. Es menester aprovechar y preparar gente con los conocimientos teóricos y tecnológicos necesarios para enfrentar el desafío que implica el cambio, para dejar de ser sólo consumidores de tecnología.

- Las aplicaciones quedan como programas de código abierto, para que otros programadores puedan hacerle mejoras y adaptarlas a sus necesidades, por ejemplo, migrarlo a POO, migrarlo a otras plataformas (Mac OS, Windows), análisis de encabezados de extensión específicos, reensamblado de paquetes, etc.

- Este trabajo de tesis puede servir como pauta para el desarrollo y la creación de un sitio de IPv6 en nuestra Universidad. En el momento de tener la infraestructura necesaria nuestra Universidad podrá realizar pruebas de comunicación con las diferentes instituciones que ya trabajan este protocolo, y en un futuro incorporarla al 6bone.

Referencias

RFC's

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

[2] R. Hinden, S. Deering, IP Version 6 Addressing Architecture , RFC 2373, Julio 1998

[3] A. Conta, S. Deering, Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) , RFC 2463, Diciembre 1998

[4] R. Gilligan, S. Thomson, J. Bound, W. Stevens, Basic Socket Interface Extensions for IPv6 , RFC2553 , Marzo 1999

[5] W. Stevens, M. Thomas, Advanced Sockets API for IPv6, RFC 2292, Febrero 1998

[6] Huitema, Chair, IAB Recommendation for an Intermediate Strategy to Address the Issue of Scaling, RFC 1481, Julio 1993

[7] Internet Engineering Steering Group, R. Hinden Editor, Applicability Statement for the Implementation of Classless Inter-Domain Routing (CIDR), RFC 1517, Septiembre 1993

[8] Y. Rekhter, T. Li, An Architecture for IP Address Allocation with CIDR, RFC 1518, Septiembre1993

[9] V. Fuller, T. Li, J. Yu, K. Varadhan, Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy, RFC 1519, Septiembre 1993

[10] IANA, ISI, Class A Subnet Experiment, RFC 1797, Abril 1995

[11] B. Manning, Class A Subnet Experiment Results and Recommendations, RFC 1879, Enero 1996

[12] B. Leiner, Y. Rekhter, The MultiProtocol Internet, RFC 1560, Diciembre 1993

[13] R. Pethia, S. Crocker, B. Fraser, Guidelines for the Secure Operation of the Internet, RFC 1281, Noviembre 1991

[14] R. Braden, D. Clark, S. Crocker, C. Huitema, Report of IAB Workshop on Security in the Internet Architecture, RFC 1636, Junio 1994

[15] P. Metzger, W. Simpson, IP Authentication using Keyed MD5, RFC 1828, Agosto 1995

[16] P. Karn, P. Metzger, W. Simpson, The ESP DES-CBC Transform, RFC 1829, Agosto 1995

[17] P. Almquist, Type of Service in the Internet Protocol Suite, RFC 1349, Julio 1992

[18] D. Eastlake, III, Physical Link Security Type of Service, RFC 1455, Mayo 1993

[19] S. Symington, D. Wood, M. Pullen, Modeling and Simulation Requirements for IPng, RFC 1667, Agosto 1994

[20] C. Mills, D. Hirsh, G. Ruth, INTERNET ACCOUNTING: BACKGROUND, RFC 1272, Noviembre 1991

[21] N. Brownlee, Accounting Requirements for IPng, RFC 1672, Agosto 1994

[22] M. Taylor, A Cellular Industry View of IPng, RFC 1674, Agosto 1994

[23] W. Simpson, IPng Mobility Considerations, RFC 1688, Agosto 1994

[24] R. Droms, Dynamic Host Configuration Protocol, RFC 1541, Octubre 1993

[25] S. Bellovin, Security Concerns for IPng, RFC 1675, Agosto 1994

[26] M. McGovern, R. Ullmann, CATNIP: Common Architecture for the Internet, RFC 1707, Octubre 1994

[27] Ross Callon, TCP and UDP with Bigger Addresses (TUBA), A Simple Proposal for Internet Addressing and Routing, RFC 1347, Junio 1992

[28] D. Piscitello, Assignment of System Identifiers for TUBA/CLNP Hosts, RFC 1526, Septiembre 1993

[29] D. Piscitello, Use of ISO CLNP in TUBA Environments, RFC 1561, Diciembre 1993

[30] R. Hinden, Simple Internet Protocol Plus White Paper, RFC 1710, Octubre 1994

Páginas web

[31] ANEXO 1: IP versión 6
"http://www.unet.edu.ve/materias/electronica/ing_redes/ANEXOS/anexo1.html"

[32] IPng (IPv6) "http://www.argo.es/~jcea/proyecto/ip6.htm"

[33] LA NUEVA VERSIÓN DE IP
"http://members.tripod.com.pe/r_marca/pagina5.htm"

[34] Protocolo IP (versión 4 y 6)
"http://www.disc.ua.es/asignaturas/rc/trabajos/ip/22.html"

[35] La transmisión de información en Internet
"http://tejo.usal.es/~nines/d.alumnos/tcpip/parte2.html"

[36] Evolución de Internet
"http://www.tid.es/presencia/publicaciones/comsid/esp/articulos/vol72/internet/internet.html"

[37] IPv4 vs IPvs6 "http://tejo.usal.es/~nines/d.alumnos/ipv6/"

[38] Tutorial y descripción técnica de TCP/IP
"http://www.ulpgc.es/otros/tutoriales/tcpip/3376c216.html",
"http://www.cicei.ulpgc.es/gsi/tut_tcpip/3376fm.html"

[39] Peter Bieringer's IPv6 & Linux - HowTo - Main
"http://www.bieringer.de/linux/IPv6/ "

[40] "http://www.monterey.edu/students/sz/stauffernataliej/world/index.htm "

[41] BSTJ version of C.ACM Unix paper
"http://cm.bell-labs.com/cm/cs/who/dmr/cacm.html"

[42] IPng Current Specifications
"http://playground.sun.com/pub/ipng/html/specs/specifications.html"

[43] IPv6 Related Specifications "http://www.ipv6.org/specs.html"

[44] IPv6: La siguiente generación (ipng)
"http://www.consulintel.es/Html/ForoIPv6/Documentos/IPv6%20-%20La%20Nueva%20Generaci%C3%B3n.pdf"

[45] IPv6 and the Future of the Internet "http://www.sun.com/software/white-papers/wp-ipv6/ipv6wp.pdf"

[46] Guía de Procedimientos Administrativos de la Infraestructura de Internet
"http://twin.uoregon.edu/~joelja/present/nsrc-esp.PDF"

[47] Direcciones IP se acabarán en 2005 "http://www.diarioti.com/noticias/2002/feb2002/15195763.html"

[48] IPV6 México "http://www.ipv6.itesm.mx/"

[49] IPv6 Resource Centre "http://www.cs-ipv6.lancs.ac.uk/ipv6/"

[50] IDC Home Page "http://www.idc.com"

[51] LinuxPPP http://www.linuxppp.com"

[52] Red Hat Linux "http://www.redhat.com"

[53] Linux Users Group Argentina (Portal de la Comunidad Argentina de Linux) http://www.linux.org.ar/

[54] GNU's Not Unix! - the GNU Project and the Free Software Foundation (FSF) "www.gnu.org"

[55] Webpage of Kame Project "http://www.kame.net/"

[56] Home page for v6 working group, WIDE project http://www.v6.wide.ad.jp

[57] IPv6 Forum http://www.ipv6forum.org

[58] TAHI Project http://www.tahi.org

[59] USAGI Project - Linux IPv6 Development Project "http://www.linux-ipv6.org/"

Libros

[60] Comer, Douglas E. "Redes globales de información con Internet y TCP/IP, Principios básicos, protocolos y arquitectura", Tercera Edición, Prentice Hall, 1996

[61] López Ángel, Novo Alejandro. "Protocolos de Internet. Diseño e implementación en sistemas UNIX", Primera Edición, Alfaomega ra-ma

Apéndice A. Manual del usuario

En este apartado se escribe el manual de usuario del programa de análisis de paquetes IPv6. Esta aplicación captura paquetes que fluyen en una red que funciona bajo el protocolo IPv6 y muestra un análisis de los campos del encabezado, también realiza un análisis básico del encabezado ICMPv6 para ejemplificar en el código fuente como se analizan los encabezados de extensión.

Requerimientos del sistema.

El sistema requiere para su funcionamiento:

1. Estar conectado a una red en la que pueda transmitir y recibir paquetes IPv6.
2. Dos estaciones de trabajo corriendo RH 7.2, RH 6.2, etc.
3. Capacidad para manejar de direcciones de IPv6.
4. Compilador GNU de C.
5. Biblioteca de desarrollo libpcap.
6. Tcl/tk.

Instalación.

Los pasos a seguir para la instalación son los siguientes

1. Para realizar la instalación es necesario abrir una sesión en la estación de trabajo.
2. Descomprimir el archivo

   tar zxvf path_del_archivo/snifo.tar.gz
3. Cambiarse al directorio creado

   cd snifo

4. Compilar

make

Ejecución del programa

Para ejecutar el programa debe ser root[1], ya que el sistema trabaja directamente con la interfaz de red y sólo root la puede manipular.

El programa tiene dos modos de operación, en modo texto y con la interfaz gráfica. Se inicia la explicación para el modo texto. Para capturar paquetes en modo texto, es necesario estar en el directorio donde se descomprimió y compilo el código fuente, la sintaxis para capturar es la siguiente *./captura #_paquetes path+nombre filtro* como se muestra a continuación

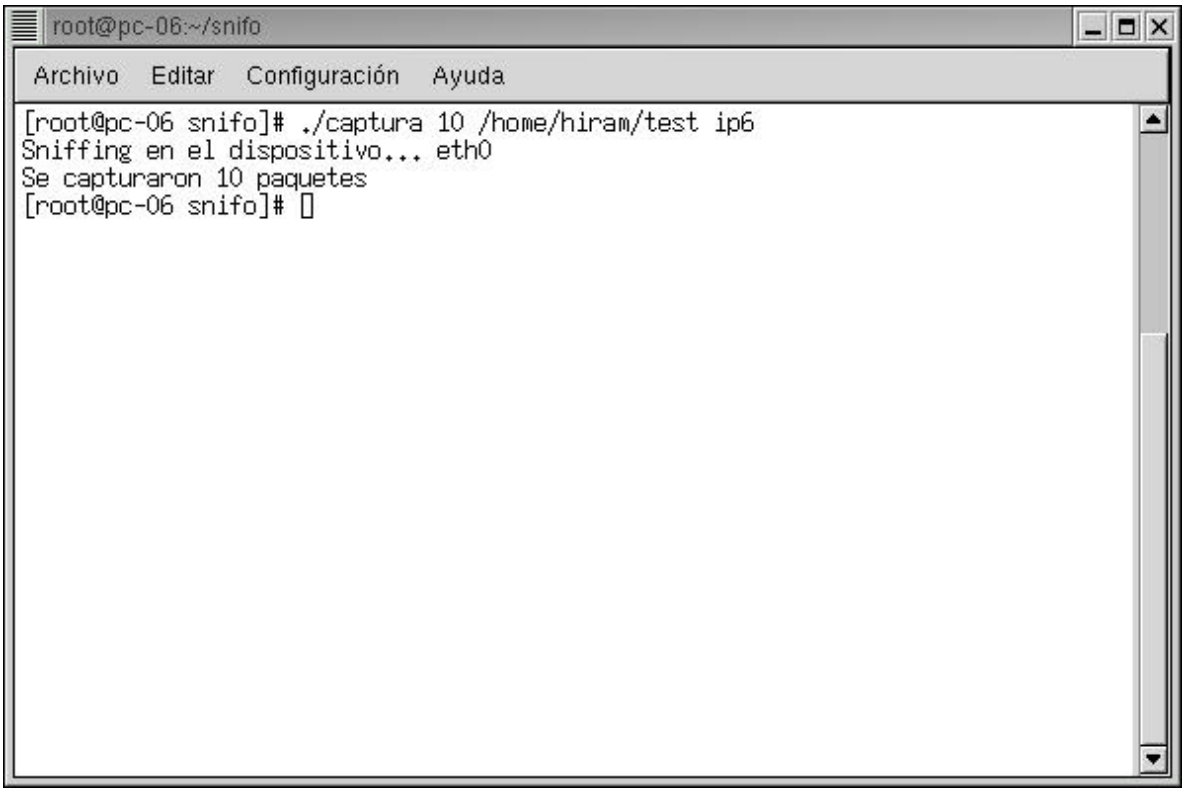

Figura 1. Ejemplo del programa de captura en modo texto

En la figura 1, se muestra un ejemplo de una ejecución del programa de captura y muestra dos mensajes de salida en la pantalla del monitor, uno de ellos se refiere a la cadena con que se identifica la interfaz de red, la otra es el número de paquetes que se capturaron. La captura se guarda en el archivo indicado en el segundo parámetro con la extensión ".ip6".

[1] Super usuario

Si falta alguno de los parámetros o si alguna de las operaciones falla, el programa aborta y envía un mensaje de error.

Para el análisis de paquetes en modo texto, la sintaxis es la siguiente *./analiza path+nombre + .ip6* como se muestra a continuación

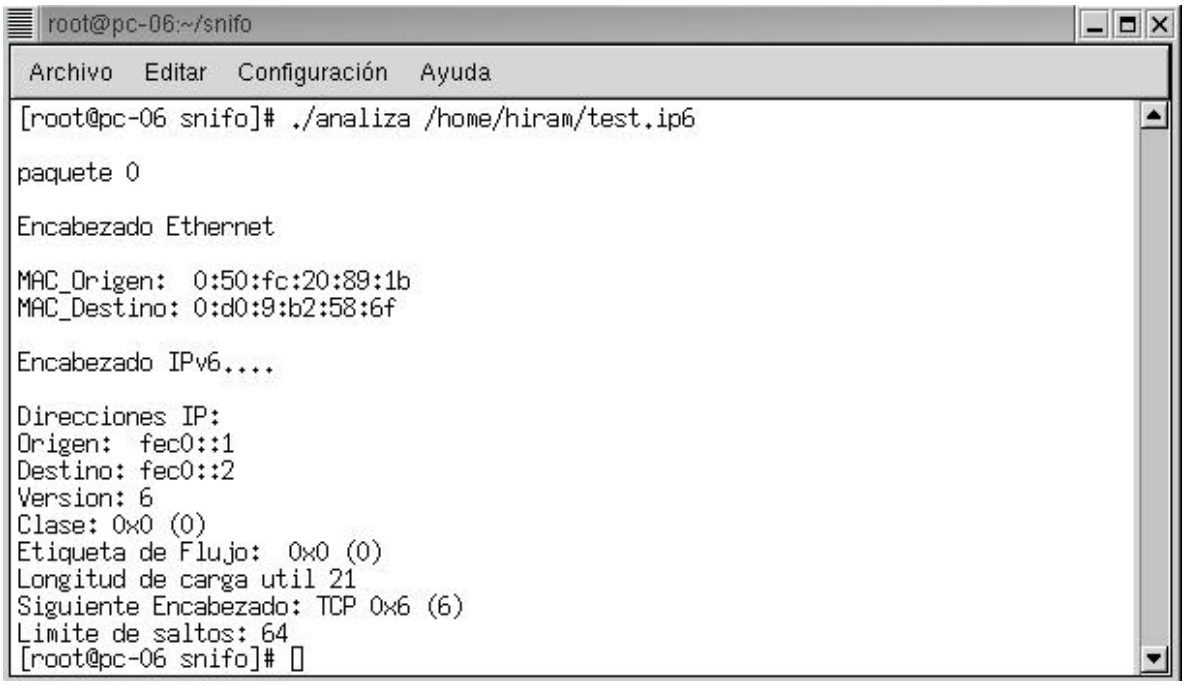

Figura 2. Ejemplo del programa de análisis en modo texto

En la figura 2, se muestra la salida del programa de análisis, presenta en la pantalla los encabezados Ethernet e IPv6, en caso de que el paquete a analizar sea un ICMPv6, muestra también un análisis básico.

En caso de que el número de parámetros sea incorrecto o no se pueda abrir el archivo, aborta y manda un mensaje de error.

Si bien es cierto que la salida del programa se pierde si es de más de 24 renglones, la salida se puede redireccionar a un archivo con la siguiente sintaxis ./analiza /path+nombre + .ip6 >nombredearchivo.

Este modo de operación de los programas es cuando las estaciones de trabajo no cuentan con un servidor X[2].

Para ejecutar el programa en la interfaz gráfica solamente es necesario escribir *./ipv6.tcl* y presenta la siguiente pantalla

[2] En sistemas Unix/Linux se refiere a el gestor de ventanas

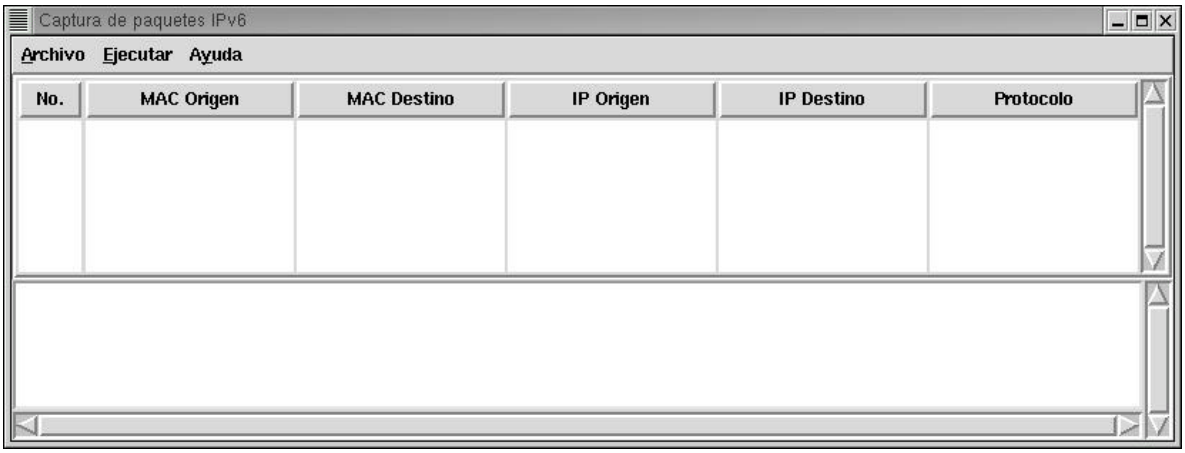

Figura 3. Pantalla principal del programa en modo gráfico

Esta interfaz muestra en el primer frame una serie de columnas donde se muestra el número de paquete que se está analizando, las direcciones MAC origen y destino, así como las direcciones IPv6.

El programa tiene tres menús, en Archivo se despliega un menú con los siguientes elementos

- Abrir - abre archivos de la captura reciente, archivos de anteriores capturas, hechos en modo texto o hechos por tcpdump.
- Guardar – guarda en un archivo de texto el análisis de los paquetes.
- Salir – sale del programa.

En Ejecutar se permite capturar y en ayuda se presenta una pequeña ayuda del sistema.

Para capturar paquetes es necesario entrar al menú Ejecutar y seleccionar la opción Capturar y muestra la siguiente pantalla donde se introducen los parámetros que se le pasan al programa de captura.

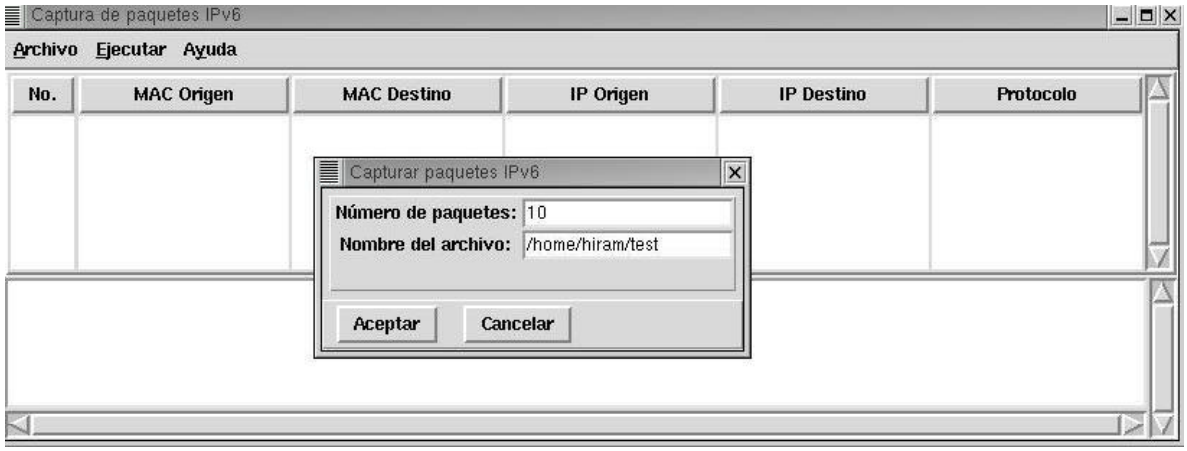

Figura 4. Pantalla para introducir parámetros al programa de captura

El programa captura los paquetes y cuando termina regresa el control para realizar alguna otra operación.

Para abrir un archivo para su análisis se abre el menú archivo y se selecciona abrir, lo cual presenta la siguiente pantalla

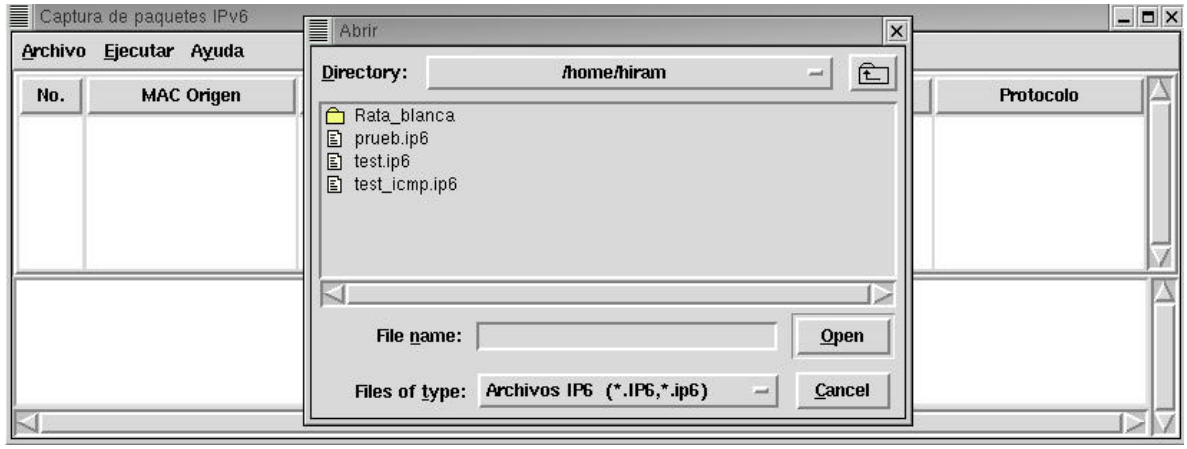

Figura 5. Cuadro de diálogo para abrir un archivo

El programa realiza el análisis y muestra la información separada en los frames que componen a la interfaz gráfica, como se muestra en la siguiente figura

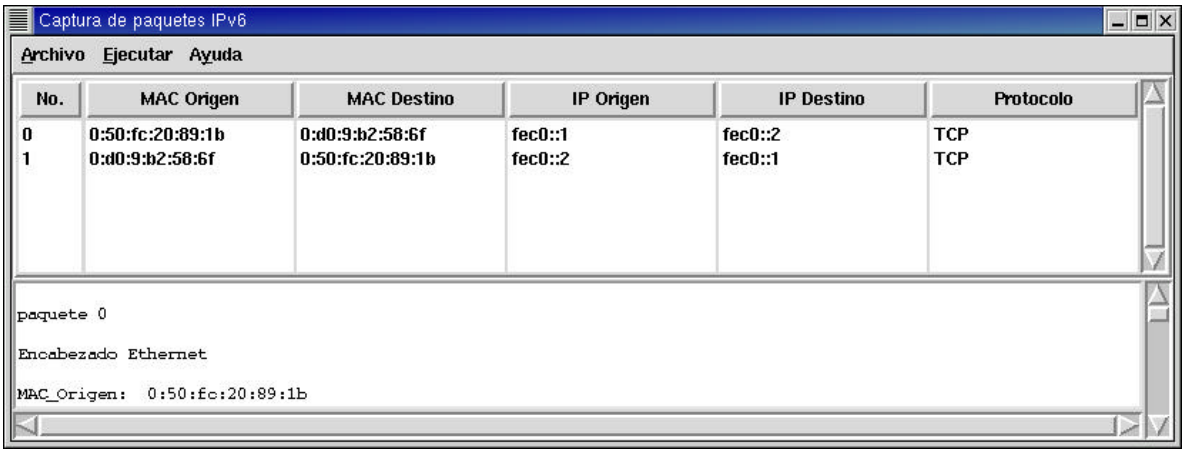

Figura 6. Pantalla que muestra el análisis de paquetes

El análisis se puede guardar en un archivo de texto para su posterior análisis si entra al menú archivo y selecciona guardar como se muestra en la siguiente figura

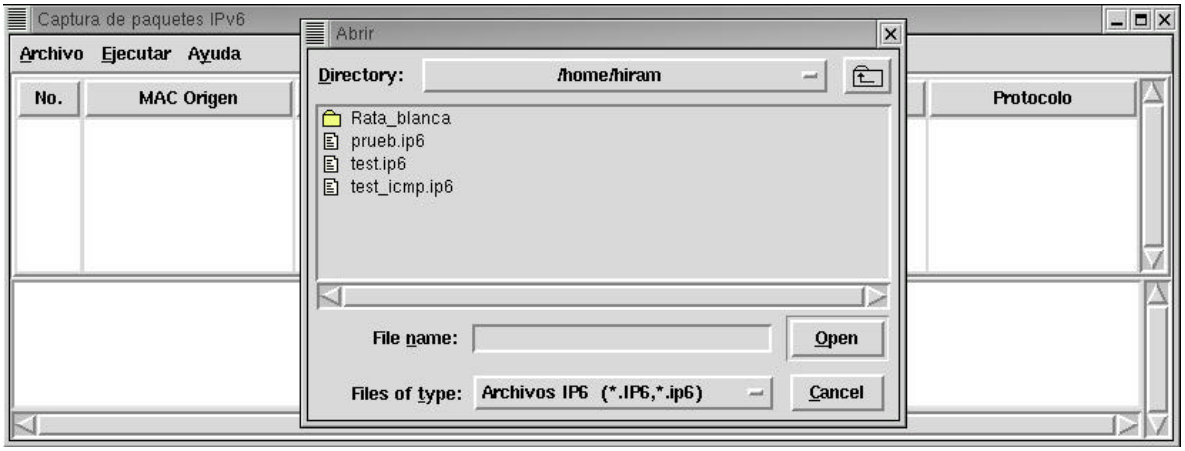

Figura 7. Pantalla que muestra el cuadro de dialogo para guardar el análisis

La opción para salir del programa se entra al menú Archivo y se selecciona salir.

Apéndice B. Artículo: Direcciones IP se acabarán en 2005

El siguiente articulo es una publicación de **www.diarioti.com** en España fechado en Febrero del 2002

Preocupación en la Unión Europea:

"Direcciones IP se acabarán en 2005"

(25.02.2002): Una antigua amenaza contra Internet aparece en versión renovada. La Unión Europea advierte que en sólo tres años más se habrán acabado las direcciones IP (Protocolo de Internet), con el consiguiente colapso del sistema.

Ya en 1996-97, una serie de expertos en Internet señalaron que el desarrollo de Internet habría de estancarse ya que el sistema pronto saturaría su capacidad de generación de direcciones. En la oportunidad, los expertos señalaron que la única forma de dar continuidad a la red sería cambiando IPv4 (Protocolo de Internet, versión 4) por IPv6. IPv4 ya tiene 20 años de uso.

Desde entonces, el tema quedó confinado a los círculos de expertos y desarrolladores de IPv6, hasta que ahora la UE decidió abordarlo nuevamente. Junto con recordar la gravedad del problema y los riesgos presentes, la entidad recomienda a las autoridades nacionales de “los 15” y al empresariado comunitario acelerar los trabajos tendientes a la implantación del próximo protocolo.

La situación se hace más urgente ahora, que no sólo los PC están conectados a Internet, sino también numerosos nuevos artículos electrónicos como Asistentes Digitales Personales y teléfonos móviles. En el mediano plazo, la situación podría ser aún más precaria, cuando artículos como refrigeradores y lavadoras comiencen a ser conectados masivamente a Internet. Ello, debido a que cada aparato necesitará contar con su propia dirección IP al conectarse a la red.

La mayoría de los sistemas operativos y los equipos para Internet ya cuentan con soporte incorporado para IPv6, pero la transición generalizada al nuevo protocolo será una tarea gigantesca.