Universidad Tecnológica de la Mixteca

ANALIZADOR Y VISUALIZADOR DE PAQUETES ICMP, ARP Y RARP

TESIS

PARA OBTENER EL TÍTULO PROFESIONAL DE:

INGENIERO EN COMPUTACIÓN

PRESENTA:

MARA IVETT RAMOS GONZÁLEZ

ASESOR:

M.C. GABRIEL GERÓNIMO CASTILLO

HUAJUAPAN DE LEÓN, OAXACA OCTUBRE DE 2001

A mis padres, Rafael y Ma. Inés

Por tener confianza en mí y haberme
apoyado incondicionalmente para
poder llegar hasta aquí.

A mis hermanos, Rafael y Paty

Porque a pesar de la distancia
estuvimos unidos siempre.

Agradecimientos.

Agradezco a los Profres. Gabriel y Everth por haberme asesorado y apoyado durante el desarrollo de este trabajo de tesis, y a mis sinodales los Profres. Luis Zarza, Francisco de Asís y Gerardo Cruz por haber brindado parte de su tiempo para la revisión de la misma.

A Carlos por brindarme su apoyo y haberme alentado para seguir adelante durante todo este tiempo.

A todo el equipo de Yukunitzá por haber sabido esperar y brindarme su amistad incondicional.

A todas las personas que de algún modo contribuyeron en el desarrollo de este trabajo.

Contenido

Lista de Figuras

Lista de Tablas

INTRODUCCIÓN

El grupo de protocolos TCP/IP han desempeñado un papel importante desde sus inicios hasta nuestros días, ya que han permitido la comunicación entre redes de diferentes características tanto de hardware como de software.

El crecimiento en el número de máquinas conectadas a una red y la seguridad en la transmisión de la información fueron dos aspectos que no se consideraron al diseñar los protocolos TCP/IP. De los dos aspectos mencionados, el segundo es el más importante, ya que cada vez son más los ataques que se realizan a sistemas conectados tanto a redes locales como a Internet. Existen muchas herramientas disponibles en Internet que permiten realizar dichos ataques por lo que surge la necesidad de que los administradores de red[1] estén siempre pendientes del tráfico que circula por su red. Existen también, herramientas disponibles en Internet que permiten el monitoreo de redes, la mayoría de estas herramientas tienen como única función capturar el tráfico y presentarlo al administrador en modo texto, algunas tienen opciones para delimitar el tipo de tráfico que se desea capturar, en otras se puede capturar todo el tráfico que circula por la red y saber hasta el final de su ejecución la cantidad de tráfico de cada tipo que ha circulado. Al estar monitoreando una red es importante que el administrador pueda visualizar el tipo y cantidad de tráfico que circula, ya que esto le permitirá tomar decisiones adecuadas para mantener el buen funcionamiento de la misma.

Para que un administrador pueda tener un control completo sobre su red necesita hacer uso de un conjunto de herramientas, esto hace que surja la necesidad de crear un sistema integral de monitoreo de redes que cuente con todas las funciones necesarias para que un administrador no tenga que trabajar con diferentes herramientas, sino que en una sola encuentre todas las funciones para mantener un buen desempeño de su red.

[1] Las palabras administrador de red y administrador se utilizarán indistintamente en este trabajo de tesis.

El sistema integral de monitoreo de redes estará formado por un conjunto de herramientas que se encargarán de capturar, visualizar y graficar los diferentes paquetes que circulan por una red Ethernet, mostrará información importante como: conexiones abiertas, puertos usados, páginas visualizadas, contraseñas utilizadas y datos transportados. El sistema será capaz de trabajar con el conjunto de protocolos TCP/IP versión 4 y versión 6, este sistema será implementado para trabajar en plataforma Linux, y será una aportación más a la comunidad de desarrolladores de software libre.

Los objetivos de este trabajo de tesis son los siguientes:

- Construir los módulos encargados de capturar y visualizar el tráfico ICMP[2], ARP[3] y RARP[4] de una red funcionando bajo IPv4.
- Generar las gráficas correspondientes a dicho tráfico.
- Describir los tipos de ataques utilizando los protocolos ICMP, ARP y RARP.
- Documentar y plantear las bases para la actualización de la herramienta para redes IPv6[5].

La herramienta implementada en este trabajo de tesis ha sido desarrollada utilizando el lenguaje C y Tcl/Tk. Esta herramienta pudo realizarse en cualquier lenguaje que brinde opciones de interfaces gráficas, tales como GTK, Perl/TK, QT, etc., la inclinación por el lenguaje Tcl/Tk se debió a la facilidad de aprendizaje, el gran uso en el desarrollo de herramientas de red, la elección de CISCO y Motorola para incluirlos en sus pruebas y desarrollos de equipos, y la portabilidad a diferentes sistemas operativos tales como Windows, UNIX y Macintosh.

El contenido de la tesis está dividido en cuatro capítulos, el primero de ellos muestra las características de los protocolos IP versión 4, dando énfasis a los protocolos ICMP, ARP y RARP dado que son los protocolos utilizados para el desarrollo de la herramienta. En el capítulo dos se muestran algunos tipos de ataques implementados con el uso de algunos de los campos de los formatos ICMP, ARP o RARP, se describen y se muestran las salidas de algunas herramientas libres para realizar y verificar dichos ataques. En el capítulo tres se muestra el diseño y desarrollo de la herramienta, así como los resultados obtenidos. El capítulo cuatro tiene como objetivo mostrar las características de los nuevos protocolos de Internet IPv6, ICMPv6[6] y ND[7], y además describir los cambios necesarios para adaptar la herramienta desarrollada a los protocolos IPv6.

---

[2] *Internet Control Message Protocol.*- Protocolo de Control de Mensajes de Internet
[3] *Address Resolution Protocol.*- Protocolo de Asociación de Direcciones
[4] *Reverse Address Resolution Protocol.*- Protocolo de Asociación de Direcciones por Réplica
[5] *Internet Protocol Version 6.*- Protocolo Internet Versión 6
[6] *Internet Control Message ProtocolVersion 6.*- Protocolo de Control de Mensajes de Internet Version 6
[7] *Neighbor Discovery.*- Descubrimiento de Vecinos

Capítulo 1

PROTOCOLOS TCP/IP

En este capítulo se describe la estructura y funciones de los protocolos TCP/IP, principalmente se hablará acerca de protocolos ICMP, ARP y RARP. También se dará una breve explicación sobre la forma de identificar los elementos que pertenecen a una red.

1.1 RESEÑA HISTÓRICA

Los protocolos TCP/IP son un grupo de estándares que especifican la forma en que se comunican las máquinas, al igual que las reglas para interconectar redes y la manera de rutear el tráfico.

A mediados de los años setenta, ARPA[1] inició un proyecto de investigación sobre las tecnologías de comunicación en redes de diferentes características. Era la principal agencia en dar recursos para la investigación en redes de conmutación de paquetes, además de ser pionera de muchas ideas sobre el tema con su red ARPANET [37]. ARPA llamó la atención de muchos investigadores por su disponibilidad en recursos para la investigación, sobre todo de aquellos que tenían experiencia utilizando conmutación de paquetes en ARPANET.

[1] *Advanced Research Projects Agency*.- Agencia de Proyectos de Investigación Avanzada

En 1980, ARPA empezó a convertir las máquinas conectadas a su red de investigación en máquinas con el nuevo protocolo TCP/IP, dando inicio de esta forma a la Internet global y convirtiendo a ARPANET en la columna vertebral de la nueva Internet.

En enero de 1983, la Oficina del Secretario de Defensa ordenó que todas las máquinas conectadas a redes de largo alcance hicieran uso de TCP/IP, y la Agencia de Comunicación de la Defensa (DCA), dividió ARPANET en dos redes, una para la investigación que conservó el nombre de ARPANET y otra para la comunicación militar conocida como red militar MILNET.

ARPA llegó a más del 90% de los departamentos de investigación universitarios, al proporcionar fondos a Bolt Beranek de Newman, Inc. (BBN) para que llevara a cabo la implementación de sus protocolos TCP/IP en la utilización de UNIX, además de otorgar fondos también a Berkeley para integrar los protocolos en su sistema de distribución de software.

El éxito que tuvo la tecnología TCP/IP pronto se expandió entre investigadores de otras áreas, y a finales de los años ochenta estaban conectadas cientos de redes individuales localizadas en Estados Unidos y Europa.

El uso de protocolos TCP/IP no se ha limitado a instituciones de gobierno, también grandes, medianas y pequeñas empresas de diferentes ramos se han conectado a Internet, así como en los últimos años lo han hecho las casas habitación.

Algunas de las características que hicieron popular a TCP/IP son [37]:

- Independencia del fabricante del hardware
- Soporte de múltiples tecnologías
- Funcionamiento en máquinas de cualquier tamaño
- Estándar de Estados Unidos desde 1983

1.2 ESTRUCTURA DE TCP/IP

Los protocolos de red están normalmente desarrollados en capas, donde cada una es responsable de una fase dentro de la comunicación. El grupo de protocolos TCP/IP consiste de diferentes protocolos que corren en las capas del modelo OSI[2]. El corazón del grupo de

[2] *Open Systems Interconnection.*- Interconexión de Sistemas Abiertos

protocolos TCP/IP está en la capa 3 y 4, como se puede observar en la figura 1.1, ya que en estas capas se ejecutan los protocolos TCP[3] e IP[4] [41].

Figura 1.1. Relación entre el modelo OSI y el modelo TCP/IP

El protocolo TCP/IP es considerado como un sistema de cuatro capas (figura 1.2), las cuales tienen las siguientes funciones [41]:

1. La Capa de Enlace incluye el controlador de la interfaz de red en el sistema operativo y la interfaz de red correspondiente a la máquina. Juntos manejan todos los detalles de la interfaz del hardware con el cable.

2. La Capa de Red maneja el movimiento de los paquetes en la red. El ruteo de paquetes, por ejemplo, se lleva a cabo en esta capa. IP, ICMP e IGMP[5] trabajan en esta capa.

3. La Capa de Transporte provee un flujo de datos entre dos máquinas, para la capa de aplicación. Dentro del grupo de protocolos TCP/IP hay dos diferentes protocolos de transporte: TCP y UDP[6]

[3] *Transmisión Control Protocol.*- Protocolo de Control de Transmisión
[4] *Internet Protocol.*- Protocolo Internet
[5] *Internet Group Management Protocol.*- Protocolo de Administración de Grupos de Internet
[6] *User Datagram Protocol.*- Protocolo de Datagrama de Usuario

TCP provee un flujo confiable de datos entre dos máquinas. Tiene que ver con aspectos, tales como dividir los datos en paquetes de tamaños adecuados, reconocer los paquetes recibidos y configurar retardos durante el envío de los paquetes. Con esto, la Capa de Aplicación puede ignorar todos estos detalles.

UDP, provee servicios más simples a la Capa de Aplicación. Envía paquetes de datos llamados datagramas desde una máquina a otra, pero no garantiza que lleguen a su destino.

4. La Capa de Aplicación maneja los detalles de la aplicación en particular. Hay muchas aplicaciones que proveen los siguiente servicios:
   - Telnet para acceso remoto.
   - FTP[7] para almacenar y recuperar archivos.
   - SMTP[8] para correo electrónico.
   - SNMP[9], entre otros.

4	telnet, ftp, e-mail, etc.	Aplicación
3	TCP, UDP	Transporte
2	IP, ICMP, IGMP	Red
1	Controlador de la interfaz de red e interfaz de red, ARP, RARP	Enlace de Datos

Figura 1.2. Capas del grupo de protocolos TCP/IP

1.3 DIRECCIONES DE INTERNET

Cuando en un sistema de comunicaciones una máquina se puede comunicar con cualquier otra máquina, se dice que es un sistema abierto de comunicaciones. Para lograr lo anterior el sistema necesita un método aceptado de manera global para identificar cada máquina que se conecte a él.

Los identificadores de cada máquina se clasifican como nombres, direcciones o rutas. Un nombre especifica lo que es un objeto, una dirección en dónde se encuentra el objeto, y la ruta cómo llegar a ese objeto.

---

[7] *File Transfer Protocol.*- Protocolo de Transferencia de Archivos
[8] *Simple Mail Transfer Protocol.*- Protocolo Simple de Transferencia de Correo
[9] *Simple Network Management Protocol.*- Protocolo Simple de Administración de la Red

Cada máquina en Internet tiene asignada una dirección de un número entero de 32 bits, llamada dirección de Internet o dirección IP. Esta dirección es la que se utiliza en todas las comunicaciones con dicha máquina.

Cada dirección es un par (*netid, hostid*), en donde *netid* es la red a la que pertenece la máquina y *hostid* es la máquina en sí. Por lo anterior, las direcciones IP se pueden utilizar para referirse a redes así como a máquinas individuales.

Una vez dada una dirección IP, se puede determinar su tipo según los tres primeros bits, de los que son necesarios sólo dos bits para distinguir entre los tres primeros tipos primarios.

Las direcciones tipo A se utilizan para las redes que tienen más de 2^{16} máquinas (65,536), éstas asignan 7 bits al campo *netid* y 24 bits al campo *hostid*. Las direcciones tipo B, se utilizan para redes que tiene entre 2^8 (256) y 2^{16} máquinas, éstas asignan 14 bits al campo *netid* y 16 bits al *hostid*. Finalmente, las direcciones tipo C, se asignan en redes que tienen menos de 2^8 máquinas, donde asignan 21 bits al campo *netid* y sólo 8 bits al *hostid*. La tabla 1.1 muestra las diferentes clases, con el rango de valores decimales con punto que corresponde a cada tipo de dirección IP.

Las direcciones IP se escriben como cuatro enteros decimales separados por puntos, por lo tanto, la siguiente dirección de 32 bits:

11000000 01100100 10101010 00000001

se escribe de la siguiente forma:

192.100.170.1

Tabla 1.1. Rango de valores decimales con punto para las diferentes clases de direcciones IP

Tipo	Dirección más baja	Dirección más alta
A	0.0.0.0	127.255.255.255
B	128.0.0.0	191.255.255.255
C	192.0.0.0	223.255.255.255
D	224.0.0.0	239.255.255.255
E	240.0.0.0	247.255.255.255

INTERNIC[10] es el encargado de proporcionar direcciones de red a las organizaciones que desean unirse a Internet, esto, con el fin de que las direcciones de red sean únicas.

[10] *Internet Network Information Center*.- Centro de Información de la Red Internet

1.4 PRINCIPALES PROTOCOLOS TCP/IP

1.4.1 PROTOCOLO INTERNET

El protocolo IP, define el mecanismo de entrega sin conexión y no confiable, lo cual se refiere a que los paquetes no siguen una misma ruta y pueden perderse en el camino, aunque el software hace su mayor esfuerzo por entregar todos los paquetes.

El protocolo IP también define la unidad básica para la transferencia de datos utilizada a través de Internet llamada datagrama. El software IP selecciona la ruta por la que los paquetes serán enviados, aporta especificaciones para el formato de los datos y el ruteo, y además, incluye reglas para la entrega de paquetes no confiable.

Las reglas se refieren a la forma en que las máquinas y ruteadores deben procesar los paquetes, cómo y cuándo se deben generar los mensajes de error y las condiciones bajo las cuales los paquetes pueden ser descartados.

1.4.1.1 Datagrama Internet

Un datagrama se divide en áreas de encabezado y datos como se muestra en la figura 1.3. El encabezado del datagrama contiene la dirección IP de la fuente y del destino, y un campo de tipo que identifica el contenido del datagrama.

ENCABEZADO DEL DATAGRAMA	ÁREA DE DATOS DEL DATAGRAMA

Figura 1.3. Forma general de un datagrama

En la figura 1.4 se muestra el arreglo de campos en un datagrama [1]. Como el proceso de los datagramas se da en el software, el contenido y el formato son independientes del tipo de hardware.

0	4	8	16	20	24	31
VERS	LONGE	TIPO DE SERVICIO	LONGITUD TOTAL			
IDENTIFICACIÓN			BAN-DERAS	DESPLAZAMIENTO DE FRAGMENTO		
TIEMPO DE VIDA		PROTOCOLO	SUMA DE VERIFICACIÓN DEL ENCABEZADO			
DIRECCIÓN IP DE LA FUENTE						
DIRECCIÓN IP DEL DESTINO						
OPCIONES IP (SI LAS HAY)						RELLENO
DATOS						
...						

Figura 1.4. Formato de un datagrama Internet

En donde:

- VERS, contiene la versión del protocolo IP que se utilizó para crear el datagrama,
- LONGE, contiene la longitud del encabezado,
- LONGITUD TOTAL, proporciona la longitud del datagrama IP medido en bytes, incluyendo los bytes del encabezado y los datos, y
- TIPO DE SERVICIO, conocido como TOS[11], especifica cómo debe manejarse el datagrama y está divido en 5 campos, como se muestra en la figura 1.5.

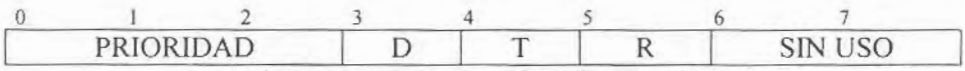


Figura 1.5. Campo Tipo de Servicio

Los primeros tres bits de precedencia especifican la prioridad del datagrama, con valores que van de 0 (prioridad normal) a 7 (control de red), permitiendo indicar al emisor la importancia de cada datagrama.

Los bits D, T y R especifican el tipo de transporte deseado para el datagrama.

- D, solicita procesamiento con retardos cortos.
- T, solicita un alto desempeño.
- R, solicita alta confiabilidad

El campo de TIPO DE SERVICIO es sólo una indicación para el algoritmo de ruteo que ayuda en la selección de una ruta hacia un destino, basándose en el conocimiento de las tecnologías de hardware disponibles en esas rutas.

Los datagramas que son enviados de una máquina a otra viajan dentro de una trama de red física. En la versión 4 del protocolo IP el tamaño máximo de un datagrama es de 2^{16} (65,535) bytes. La forma de transportar un datagrama dentro de una trama de red es conocida como encapsulado, la figura 1.6 muestra cómo el datagrama completo viaja en el área de datos de la trama de red.

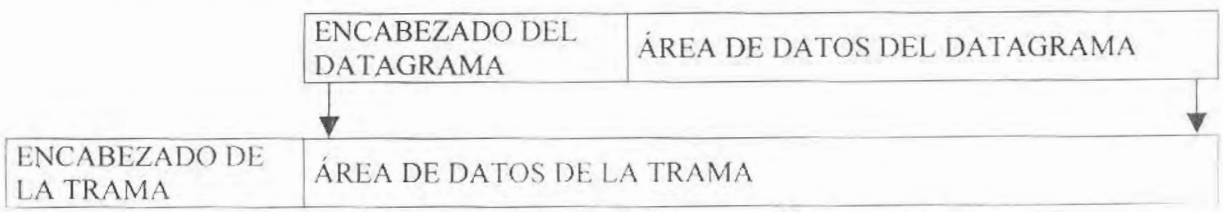


Figura 1.6. Encapsulado de un datagrama IP en una trama

[11] *Type of Service* -Tipo de Servicio

Cuando es enviado un datagrama IP encapsulado en el área de datos de la trama utiliza el valor tipo 0800_{16}.

Cada tecnología de conmutación de paquetes fija la cantidad de datos límite que puede transportar una trama física, en el caso de Ethernet la cantidad de bytes límite es de 1,500, y para FDDI es de 4,352 bytes. Esta cantidad máxima de transferencia de datos se conoce como MTU[12] de una Red. Como muchas veces los datos que se transportan viajan por diferentes redes con MTU de diferente tamaño, los datos son divididos en datagramas cuyo tamaño es seleccionado de forma que cada uno de éstos se pueda transportar a través de la red en una sola trama, a esta división se le llama fragmentación, debido a que IP representa el desplazamiento de datos en múltiplos de 8, cada fragmento debe tener como tamaño el múltiplo de 8 más cercano al MTU de la red, aunque normalmente el datagrama no es dividido en fragmentos de tamaños iguales.

Cada fragmento contiene el encabezado del datagrama original, excepto por un bit en el campo BANDERAS que indica que es un fragmento. Los fragmentos no son reensamblados durante la trayectoria que siguen, sino hasta que llegan a su destino final.

Existen tres campos en el encabezado del datagrama que sirven para controlar la fragmentación y el reensamblado, estos campos son: IDENTIFICACIÓN, BANDERAS y DESPLAZAMIENTO DE FRAGMENTO.

La función del campo IDENTIFICACIÓN es dar a conocer qué fragmentos pertenecen a qué datagramas. Cada valor en este campo debe ser único para cada datagrama.

El campo DESPLAZAMIENTO DE FRAGMENTO, especifica el desplazamiento en el datagrama original de los datos que se están transportando en el fragmento, en múltiplos de 8 bytes.

En el campo BANDERAS, los dos bits de menor orden controlan la fragmentación. El bit de orden inferior indica si el fragmento contiene datos intermedios o de la parte final del datagrama original. Revisando los campos DESPLAZAMIENTO DEL FRAGMENTO y LONGITUD TOTAL se puede saber el tamaño del datagrama original, con lo que un receptor puede saber en qué momento los fragmentos pueden ser reensamblados al datagrama original.

---

[12] *Maximum Transfer Unit* - Unidad de Transferencia Máxima

El campo TIEMPO DE VIDA sirve para controlar el tiempo que un datagrama va a estar viajando por la red, cada ruteador por el que pasa un datagrama resta el número de segundos que permanece ahí el datagrama del campo TIEMPO DE VIDA, con ello se asegura que un datagrama no va a permanecer en la red durante un tiempo indefinido, ya que cuando este se vuelve 0 el datagrama es descartado y es enviado un mensaje de error a la fuente.

El campo PROTOCOLO, especifica qué protocolo se utilizó para crear el mensaje que se está enviando en el área DATOS del datagrama.

El campo SUMA DE VERIFICACIÓN DEL ENCABEZADO, especifica la integridad de los valores del encabezado. Esta suma de verificación aplica solamente para los valores del encabezado IP y no para los datos.

Los campos DIRECCION IP DE LA FUENTE y DIRECCIÓN IP DEL DESTINO, contienen las direcciones IP de 32 bits de los datagramas del emisor y receptor involucrados en el envío de los mismos, estas direcciones nunca cambian, aunque los datagramas pasen a través de muchos ruteadores.

El campo DATOS indica el comienzo del área de datos de un datagrama.

El campo OPCIONES IP va después del campo de la dirección destino, el tamaño de este campo depende de qué opción es seleccionada, cada opción consiste en un solo byte de código de opción y un conjunto de bytes de datos para cada opción. El formato del byte de código de opción se muestra en la figura 1.7.

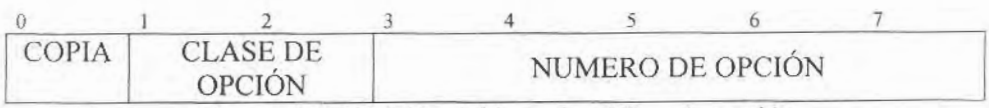


Figura 1.7. División del byte de código de opción

El byte de código de opción consiste de una bandera de 1 bit, llamada COPIA, que controla la forma en que los ruteadores tratan las opciones durante la fragmentación, si está en 1 significa que la opción se debe copiar a todos los fragmentos y si está en 0 que sólo se debe copiar en el primer fragmento y no en todos.

Los bits CLASE DE OPCIÓN y NUMERO DE OPCIÓN especifican la clase general de opción, la tabla 1.2 muestra las diferentes clases [1].

Tabla 1.2. Clases de opciones IP

CLASE DE OPCIÓN	SIGNIFICADO
0	Control de red o datagrama
1	Reservado para uso futuro
2	Depuración y medición
3	Reservado para uso futuro

Las opciones que puede tener un datagrama IP se muestran en la tabla 1.3, ésta muestra también los valores para CLASE DE OPCIÓN Y NÚMERO DE OPCIÓN.

Tabla 1.3. Las opciones posibles IP con su clase en forma numérica y los códigos de número.

CLASE DE OPCIÓN	NUMERO DE OPCIÓN	LONGITUD	DESCRIPCIÓN
0	0	-	Fin de la lista de opciones. Se utiliza si las opciones no terminan al final del encabezado.
0	1	-	No operación (se utiliza para alinear bytes en una lista de opciones).
0	2	11	Seguridad y restricciones de manejo (para aplicaciones militares).
0	3	variable	Ruteo no estricto de fuente. Se utiliza para rutear un datagrama a través de una trayectoria específica.
0	7	variable	Registro de ruta. Se utiliza para registrar el trayecto de una ruta.
0	8	4	Identificador de flujo. Se utiliza para transportar un identificador de flujo SATNET (obsoleto).
0	9	variable	Ruteo estricto de fuente. Se utiliza para establecer la ruta de un datagrama en un trayecto específico.
0	4	variable	Sello de tiempo Internet. Se usa para registrar sellos de hora a lo largo de una ruta.

Las opciones de ruteo y sello de hora ayudan a monitorear o controlar la forma en que Internet maneja las rutas de los datagramas. La figura 1.8 muestra el formato de la opción de Registro de ruta, la cual permite a la fuente crear una lista de direcciones IP y arreglarla para que cada ruteador por donde pase el datagrama añada su propia dirección IP.

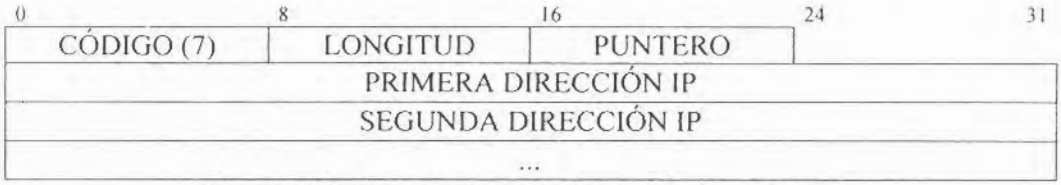


Figura 1.8. Formato de la opción de Registro de Ruta

El campo CÓDIGO contiene la opción y el número de opción para el registro de rutas. El campo LONGITUD especifica el tamaño total de la opción. El campo PUNTERO especifica el desplazamiento dentro del formato a la siguiente ranura disponible para almacenar la dirección IP. Para que el registro de ruta sea tomado en cuenta se necesita que la máquina fuente acepte la habilitación del registro de ruta y la máquina destino acepte el procesamiento de la lista resultante, de otra manera al llegar un datagrama a su destino no se tomará en cuenta la ruta registrada.

La opción de Ruta Fuente ayuda a los administradores de red a evaluar el desempeño de una red física en particular, haciendo uso de la ruta fuente el administrador puede forzar a que los paquetes viajen por una determinada red. El IP soporta dos tipos de ruteo de fuente: ruteo estricto de fuente y la otra forma conocida como ruteo incontrolado de fuente. En el ruteo estricto de fuente se especifica la ruta exacta que los datagramas van a seguir para llegar a su destino. La ruta entre dos direcciones sucesivas de la lista debe consistir en una sola red física y en el ruteo no estricto de fuente los datagramas IP pueden hacer saltos entre redes.

La figura 1.9 muestra el formato de opción de ruta fuente, el cual es muy parecido al de registro de ruta. Cada ruteador verifica si la lista está completa, de lo contrario el ruteador sigue al puntero, reemplaza la dirección IP con la dirección del ruteador y establece la ruta para el datagrama utilizando la dirección IP que obtuvo de la lista.

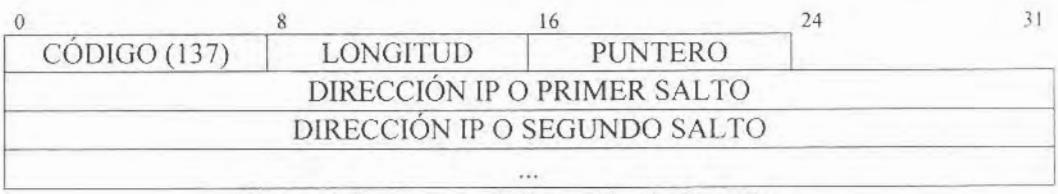


Figura 1.9. Formato de la opción de Ruta Fuente

Existe otra opción que es la de registro de sello de hora, ésta consiste de una lista vacía en la que cada ruteador que maneja un determinado datagrama introduce sus datos, cada entrada a la lista son 2 datos de 32 bits: la dirección IP del ruteador y un sello de hora de 32 bits. La figura 1.10 muestra el formato de sello de hora.

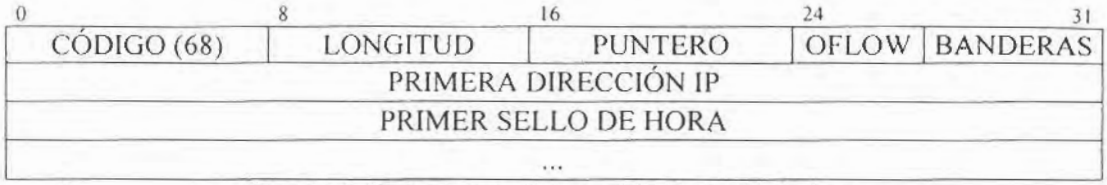


Figura 1.10. Formato de la opción de Sello de hora

Los campos LONGITUD y PUNTERO se utilizan para saber cuál es la siguiente ranura desocupada para almacenar los datos, el campo OFLOW contiene un contador entero de ruteadores que pueden no proporcionar el sello de hora si la opción fue muy pequeña. El campo BANDERAS especifica el formato de la opción y define cómo deben introducir el sello de hora los ruteadores.

La tabla 1.4 especifica la interpretación de los valores en el campo BANDERAS de la opción sello de hora.

Tabla 1.4. Significado de los valores del campo BANDERAS de la opción Sello de hora

VALOR DE LA BANDERA	SIGNIFICADO
0	Registro de sello de hora solamente; omite direcciones IP.
1	Anteponer a cada sello de hora una dirección IP (formato de la figura 1.10)
3	Las direcciones IP se especifican por el emisor; un ruteador sólo registra un sello de hora si la próxima dirección IP en la lista concuerda con la dirección IP del ruteador.

1.4.2 PROTOCOLO DE ASOCIACIÓN DE DIRECCIONES

Cuando dos máquinas requieren comunicarse entre sí, necesitan conocer sus direcciones físicas.

ARP se utiliza para la asignación de direcciones IP en forma dinámica, ya que es un protocolo de bajo nivel que deja oculto el direccionamiento físico [3].

Cuando una máquina requiere la dirección física de otra máquina, transmite por difusión un paquete especial solicitándole a esa máquina, de la que sólo conoce su dirección IP, que responda con su dirección física. Todas las máquinas conectadas a la red, reciben la solicitud, pero sólo la que reconoce su dirección IP envía una respuesta que incluye su dirección física.

Las máquinas que utilizan ARP tienen una memoria intermedia en la que almacenan las correspondencias de dirección IP con dirección física adquiridas recientemente, esto para no utilizar ARP varias veces.

Funcionalmente ARP está dividido en dos partes. La primera parte transforma una dirección IP en una dirección física al enviar un paquete y la segunda responde a solicitudes de otras máquinas.

En la primera parte del protocolo, el software ARP consulta su memoria temporal para extraer la dirección física correspondiente a la dirección IP del anfitrión a donde irá dirigido el paquete. La segunda parte del protocolo ARP es la que maneja los paquetes que llegan por medio de la red, es decir, es la encargada de responder a solicitudes ARP, así como de recibir respuestas ARP a solicitudes realizadas con anterioridad.

En la figura 1.11 se muestra cómo se transmite el mensaje ARP dentro de una trama.

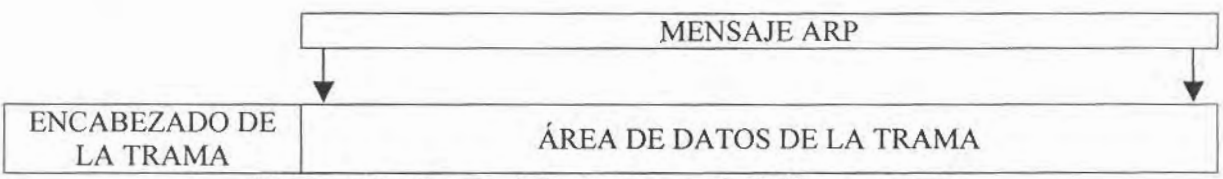


Figura 1.11. Encapsulado de un mensaje ARP en una trama

Para identificar si una trama transporta un mensaje ARP, la máquina transmisora coloca un valor en el campo tipo del encabezado de la trama y pone el mensaje ARP en el campo de datos.

En una red Ethernet, las tramas que transmiten mensajes ARP tienen en su campo tipo el 0806_{16}, el cual es un valor estándar para Ethernet.

Los datos en los paquetes ARP no tienen un encabezado con formato fijo, para que ARP se pueda utilizar en varias tecnologías, la longitud de los campos de direcciones depende del tipo de red.

En la figura 1.12, se muestra el formato de un mensaje ARP/RARP [37] que se utiliza en Ethernet.

0	8	16	24	31
TIPO DE HARDWARE		TIPO DE PROTOCOLO		
LONGH	LONGP	OPERACIÓN		
HA TRANS (bytes 0-3)				
HA TRANS (bytes 4-5)		IP TRANS (bytes 0-1)		
IP TRANS (bytes 2-3)		HA RECEPTOR (bytes 0-1)		
HA RECEPTOR (bytes 2-5)				
IP RECEPTOR (bytes 0-3)				

Figura 1.12. Formato de un mensaje ARP/RARP

En donde:

- TIPO DE HARDWARE, especifica un tipo de interfaz de hardware para el que el transmisor busca una respuesta, para Ethernet tiene un valor de 1,
- TIPO DE PROTOCOLO, especifica el tipo de dirección de protocolo de alto nivel que proporcionó el transmisor, contiene 0800_{16} para la dirección IP,
- OPERACIÓN, especifica una solicitud ARP (1), una respuesta ARP (2), una solicitud RARP (3) o una respuesta RARP (4),
- LONGH y LONGP, permiten que ARP se utilice con redes arbitrarias, pues LONGH especifica la longitud de la dirección de hardware y LONGP la longitud de la dirección del protocolo de alto nivel,
- HA TRANS e IP TRANS, en estos campos el transmisor coloca su dirección de hardware y su dirección IP respectivamente, y
- HA RECEPTOR e IP RECEPTOR, son utilizados por el transmisor para especificar la dirección IP del objetivo para el caso de ARP y la dirección de hardware del objetivo para RARP.

1.4.3 PROTOCOLO DE ASOCIACIÓN DE DIRECCIONES POR RÉPLICA

RARP se utiliza cuando una máquina desea conocer su dirección IP [4]. Cuando esto sucede la máquina envía una solicitud a un servidor RARP y espera a que éste le envíe una respuesta. Durante esta breve comunicación, las dos máquinas utilizan sus direcciones físicas de red. Para que RARP funcione bien debe existir dentro de la red por lo menos un servidor RARP.

RARP es una adaptación de ARP y ambos tienen el mismo formato para sus mensajes como se muestra en la figura 1.12. Un mensaje RARP se envía de una máquina a otra encapsulado en la porción de datos de una trama. El tipo de paquete que contiene la trama es 8035_{16} con esto se identifica que es una trama que contiene un mensaje RARP, el cual es de 28 bytes.

Debido a que RARP utiliza directamente la red física, ningún otro software distinto al software RARP cronometra o retransmite la respuesta.

Hay dos formas de asegurar que una máquina solicitante obtenga una respuesta y no haya congestionamiento en la red. Una es establecer un servidor primario para cada máquina, de esta manera se asegura que ninguno de los otros servidores RARP manden una respuesta al mismo tiempo, ya que sólo el servidor primario asignado podrá contestar aunque los demás también reciben la solicitud. La máquina solicitante cronometra el tiempo de respuesta y, si no la obtiene, transmitirá por difusión nuevamente la solicitud. Así, cuando un servidor no primario recibe una segunda solicitud, poco tiempo después de la primera, responde.

La segunda forma, es similar a la anterior, sólo que en ésta se evita que todos los servidores no primarios respondan simultáneamente al recibir una segunda solicitud, haciendo que cada máquina no primaria al recibir una solicitud calcule un retraso aleatoriamente y después envíe la respuesta. Así, en el caso de no existir ningún problema, el servidor primario responde inmediatamente y las demás respuestas se retrasan, y cuando el servidor primario no está disponible, existe un pequeño retraso en la máquina solicitante antes de recibir la respuesta.

1.4.4 PROTOCOLO DE CONTROL DE MENSAJES DE INTERNET

ICMP es un protocolo utilizado por las máquinas y los ruteadores para transportar información de control o de error hacía otras máquinas o ruteadores.

Los mensajes ICMP sólo pueden reportar los errores a la fuente original del datagrama y no especifican qué acción tomar para corregirlo. Debido a que puede ser necesario que estos mensajes viajen a través de muchas redes físicas para llegar a su destino final no es posible enviarlos directamente por un medio físico, sino que es necesario que viajen dentro de un datragrama IP, por lo que tienen dos niveles de encapsulado como se puede observar en la figura 1.13, viajan en la porción de datos de un datagrama IP, y éste a su vez viaja en la porción de datos de una trama. Los mensajes ICMP no tienen ningún tratamiento especial, viajan igual que cualquier otro paquete, por lo que también pueden llegar a perderse, sólo que existe una excepción que hace que no se generen mensajes de error sobre un mensaje ICMP.

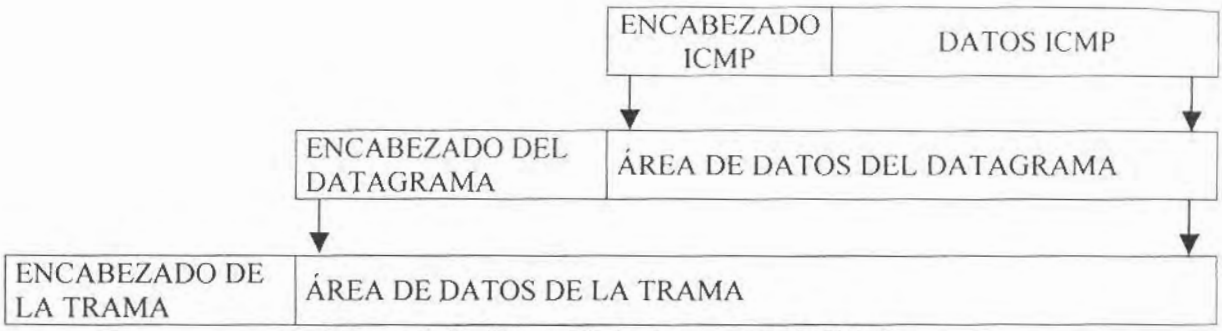


Figura 1.13. Encapsulado de un mensaje ICMP

1.4.4.1 Formato de los mensajes ICMP

Cada tipo de mensaje ICMP tiene su propio formato debido a sus diferentes funcionalidades, lo único que comparten son los tres campos con los que empiezan, que son:

- TIPO, indica el tipo de mensaje, y su valor determina el formato del resto de la cabecera,
- CÓDIGO, depende del tipo de mensaje, y es utilizado para crear un nivel adicional de jerarquía para la clasificación del mensaje, y
- SUMA DE VERIFICACIÓN, permite detectar errores en el mensaje ICMP.

Todos los mensajes ICMP que reportan errores incluyen el encabezado y los primeros 64 bits de datos del datagrama que causó el error.

Los tipos de mensaje ICMP [22] que se pueden originar se describen en la tabla 1.5.

Tabla 1.5. Tipos de mensaje ICMP

Tipo	Código	Descripción de mensaje ICMP
0	0	Respuesta de eco
3		Destino inaccesible
	0	Red inaccesible
	1	Anfitrión inaccesible
	2	Protocolo inaccesible
	3	Puerto inaccesible
	4	Se necesita fragmentación y configuración DF
	5	Falla en la ruta de origen
	6	Red de destino desconocida
	7	Anfitrión de destino desconocido
	8	Anfitrión de origen aislado
	9	Comunicación con la red de destino administrativamente prohibida

Tabla 1.5. Tipos de mensaje ICMP (Continúa)

Tipo	Código	Descripción de mensaje ICMP
	10	Comunicación con el anfitrión de destino administrativamente prohibida
	11	Red inaccesible por el tipo de servicio
	12	Anfitrión inaccesible por el tipo de servicio
	13	Comunicación administrativamente prohibida
	14	Violación de precedencia del anfitrión
	15	Anticipación de un atajo
4	0	Disminución de tasa al origen
5		Redireccionar (cambiar una ruta)
	0	Redireccionar datagramas para la red (ahora obsoleto)
	1	Redireccionar datagramas para el anfitrión
	2	Redireccionar datagramas para el tipo de servicio y la red
	3	Redireccionar datagramas para el tipo de servicio y el anfitrión
8	0	Solicitud de Eco
9	0	Notificación de ruta
10	0	Solicitud de ruta
	0	Conteo de tiempo de vida excedido
	1	Tiempo para el reensamblado de fragmentos excedido
12		Problema de parámetros en un datagrama
	0	Error en el encabezado IP
	1	Falta opción requerida
13	0	Solicitud de marca de hora
14	0	Respuesta de marca de hora
15	0	Solicitud de información (obsoleto)
16	0	Respuesta de información (obsoleto)
17	0	Solicitud de máscara de dirección
18	0	Respuesta de máscara de dirección
19	0	Reservado (para Seguridad)
20-29		Reservado para experimentos robustos
30		Trazado de ruta
31		Error de conversión de datagrama
32		Redireccionamiento de anfitrión móvil
33		IPv6 ¿Dónde estás?
34		IPv6 Estoy aquí
35		Solicitud de registro de movimiento
36		Respuesta de registro de movimiento
39		Rebote

Tabla 1.5. Tipos de mensaje ICMP (Continúa)

Tipo	Código	Descripción de mensaje ICMP
40		Photuris
	0	Reservado
	1	Índice de parámetros de seguridad desconocidos
	2	Parámetros de seguridad válidos, pero autenticación errónea
	3	Parámetros de seguridad válidos, pero desencriptación errónea

1.4.4.2 Solicitud y respuesta de eco

Los mensajes ICMP de solicitud y respuesta de eco son utilizados por las máquinas para comprobar si el destino al que van a enviar información es alcanzable o no, una máquina origen formula una solicitud de eco y la máquina destino da respuesta a esta solicitud, en el caso de que la máquina origen reciba exitosamente una respuesta de eco, quiere decir que los elementos necesarios para llevar a cabo la comunicación entre el origen y el destino están trabajando correctamente.

El formato para estos mensajes [2] se muestra en la figura 1.14.

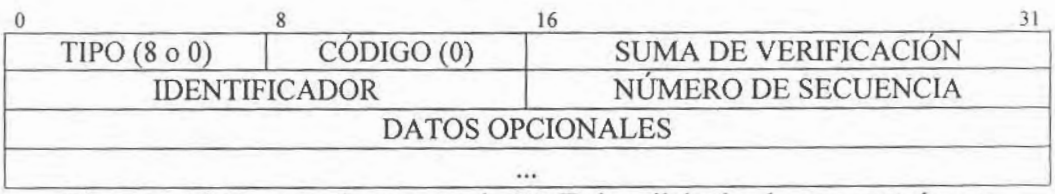


Figura 1.14. Formato de un mensaje ICMP de solicitud o de respuesta de eco

En donde:

- IDENTIFICADOR y NÚMERO DE SECUENCIA, son utilizados por el transmisor para responder a las solicitudes de eco, y
- DATOS OPCIONALES, son los datos de respuesta de eco que se regresa al transmisor.

1.4.4.3 Destino inaccesible

Este tipo de mensajes ICMP es utilizado por una máquina destino cuando el protocolo especificado en el campo número de protocolo del datagrama original no está activo en la máquina destino, o el puerto especificado está inactivo.

También es utilizado por un ruteador cuando no puede direccionar o entregar un datagrama IP, los casos en los que puede darse este tipo de error son: que el hardware esté fuera de servicio por un tiempo, se haya especificado una dirección de destino no existente, o el ruteador no tenga una ruta para la dirección destino. En ese momento el ruteador envía un mensaje ICMP y luego deja suelto el datagrama.

El formato de este mensaje se muestra [2] en la figura 1.15.

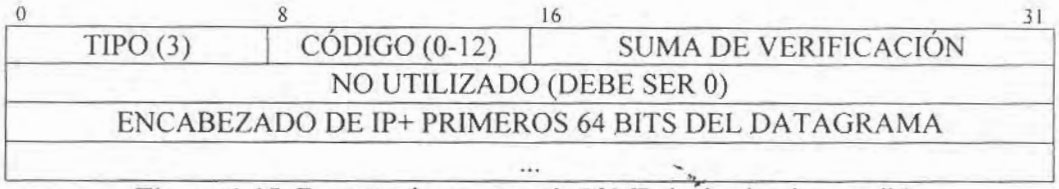


Figura 1.15. Formato de un mensaje ICMP de destino inaccesible

1.4.4.4 Ruta de origen y necesidad de fragmentación

El mensaje ICMP de necesidad de fragmentación (Código 3, Tipo 4) es enviado por un ruteador hacia la fuente cuando un datagrama tiene la opción de no fragmentar y es necesario fragmentarlo, el mensaje ICMP de falla de ruta de origen (Código 3, Tipo 5) es utilizado cuando un datagrama tiene la opción de ruta de origen con una ruta incorrecta.

1.4.4.5 Disminución de tasa al origen

Cuando los ruteadores se llegan a saturar con el tráfico se dice que hay un congestionamiento en la red, esto puede ocurrir, porque una máquina de alta velocidad genera tráfico más rápido de lo que puede ser procesada o si varias máquinas generan tráfico hacía un mismo ruteador. Para evitar los congestionamientos una máquina puede enviar un mensaje ICMP de disminución de tasa al origen, éste es una solicitud para que la máquina origen disminuya la velocidad a la que está enviando los paquetes. La máquina que recibe este tipo de mensajes ICMP no recibe ningún mensaje que le informe que deje de reducir la velocidad de transmisión, sino que lo hace al momento que deja de recibir los mensajes de disminución de tasa al origen y de ahí va aumentando la velocidad de transmisión gradualmente.

El RFC 1812 especifica que un ruteador no debería generar mensajes de disminución de tasa al origen, pero un ruteador que origina estos mensajes debe ser capaz de limitar la velocidad a la cual son generados (debido a que consume ancho de banda y es un remedio ineficaz para la congestión). Cuando un ruteador recibe un mensaje de este tipo puede ignorarlo.

El formato de estos mensajes ICMP [2] se muestra en la figura 1.16, en donde a parte de los campos TIPO, CÓDIGO, SUMA DE VERIFICACIÓN y el campo no utilizado de 32 bits, contiene un campo que incluye el prefijo del datagrama que generó el error.

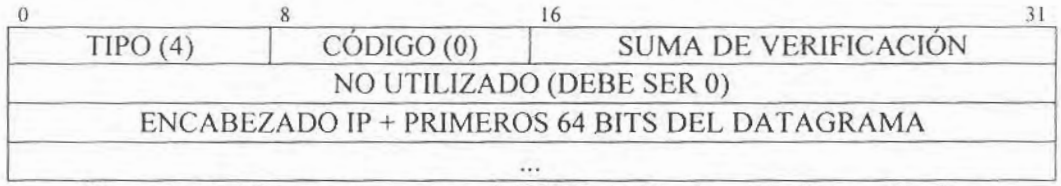


Figura 1.16. Formato de un mensaje ICMP de disminución de tasa al origen

1.4.4.6 Cambio de ruta

Este tipo de mensaje ICMP es útil porque permite que una máquina mantenga su tabla de ruteo con la mínima información posible, dejando el trabajo de actualización de sus tablas de ruteo a los ruteadores. Con lo anterior, una máquina al momento de arrancar sólo incluye en su tabla de ruteo la dirección de un ruteador, y si al querer enviar un datagrama existe una ruta mejor para dicho datagrama, el ruteador le envía a la máquina origen un mensaje ICMP, llamado redireccionar, con el cual le pide a la máquina que cambie sus rutas. Este tipo de mensajes sólo puede ser enviado de un ruteador a una máquina.

Cada mensaje de redireccionamiento además de los campos obligatorios que ya se han mencionado, contiene un campo de 32 bits conocido como DIRECCIÓN DE INTERNET DEL RUTEADOR y un campo ENCABEZADO, esto se puede ver en la figura 1.17 [2], en donde:

- DIRECCIÓN DE INTERNET DEL RUTEADOR, contiene la dirección de un ruteador que la máquina origen utilizará,
- ENCABEZADO, contiene el encabezado IP, además de los siguientes 64 bits del datagrama que generó el mensaje, y
- CÓDIGO, especifica cómo interpretar la dirección de destino, de acuerdo a la descripción de la tabla 1.5.

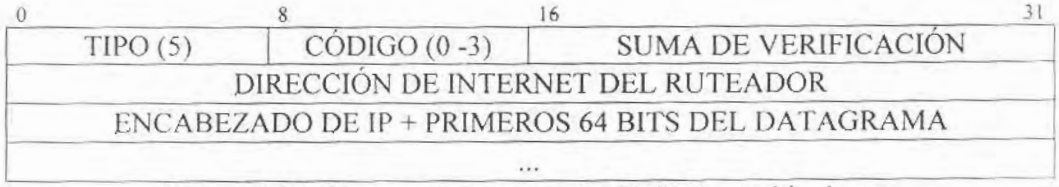


Figura 1.17. Formato de un mensaje ICMP de cambio de ruta

1.4.4.7 Mensajes de Solicitud y Respuesta de Ruta

Anteriormente, ya se había mencionado el uso de tablas de ruteo, aunque existe otra forma de especificar la ruta que debe seguir un paquete, esta forma es el uso de mensajes ICMP de solicitud y respuesta de ruta.

Cuando una máquina envía ya sea por difusión o multidifusión una solicitud de ruta, uno o más ruteadores responden con un mensaje de respuesta de ruta. Aunque, los ruteadores periódicamente envían su dirección de Internet para que las máquinas que estén escuchando en ese momento, actualicen sus tablas de ruteo.

La figura 1.18 muestra el formato del mensaje de solicitud de ruta [7], y la figura 1.19 muestra el formato de un mensaje de respuesta de ruta [7], en el cual se pueden incluir una o más direcciones.

0	8	16	31
TIPO (10)	CÓDIGO (0)	SUMA DE VERIFICACIÓN	
NO UTILIZADO (DEBE SER 0)			

Figura 1.18. Formato de un mensaje ICMP solicitud de ruta

0	8	16	31
TIPO (9)	CÓDIGO (0)	SUMA DE VERIFICACIÓN	
NÚMERO DE DIRECCIONES	TAMAÑO DE LA DIRECCIÓN DE ENTRADA	TIEMPO DE VIDA	
DIRECCIÓN DE RUTEADOR (1)			
NIVEL DE PREFERENCIA (1)			
DIRECCIÓN DE RUTEADOR (2)			
NIVEL DE PREFERENCIA (2)			
...			

Figura 1.19. Formato de un mensaje ICMP respuesta de ruta

En el mensaje de respuesta de ruta:

- NÚMERO DE DIRECCIONES, se refiere al número de direcciones que incluye el mensaje,
- TAMAÑO DE LA DIRECCIÓN DE ENTRADA, especifica el número de palabras de 32 bits para cada dirección de ruteador, este valor es siempre 2,
- TIEMPO DE VIDA, se refiere al número de segundos que la dirección puede ser considerada válida, y
- NIVEL DE PREFERENCIA, se refiere a la preferencia de esta dirección como una dirección por omisión del ruteador.

1.4.4.8 Tiempo excedido

Cada datagrama que circula por Internet tiene un campo de TIEMPO DE VIDA en el encabezado IP, el cual es configurado por la aplicación que lo envía y representa el tiempo máximo que le es permitido viajar.

El valor del campo TIEMPO DE VIDA es decrementado cada vez que el encabezado IP es procesado. El RFC 791 especifica que este campo decrementado refleja el tiempo consumido en procesar el datagrama. El valor del campo es medido en unidades de segundos. El RFC también establece que el tiempo máximo de vida puede ser configurado a 255 segundos, lo que equivale a 4.25 minutos. El datagrama es descartado cuando el campo ha llegado a cero antes de que haya llegado a su destino.

Tener mayor control en el tiempo de vida de un datagrama ayuda a prevenir que viejos duplicados lleguen después de un cierto tiempo transcurrido. Así cuando se retransmite información la cual no fue previamente entregada se puede estar seguro que el duplicado más viejo fue descartado y no interferirá con el proceso. Si un ruteador descubre que el campo TIEMPO DE VIDA en un encabezado IP que haya procesado es igual a cero, descartará el datagrama y generará un mensaje ICMP de Tiempo excedido (código 0), en el caso de que se reciba un paquete que indique que vienen más atrás de él y se exceda el tiempo de espera de los demás fragmentos, el paquete no se podrá reensamblar haciendo que se genere un mensaje ICMP de Tiempo excedido (Código 1). La figura 1.20 muestra el formato para estos mensajes.

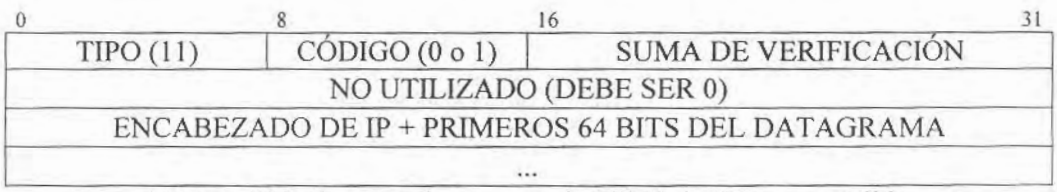


Figura 1.20. Formato de un mensaje ICMP de tiempo excedido

1.4.4.9 Parámetros incorrectos

Los mensajes ICMP que reportan parámetros incorrectos, sólo se utilizan cuando ninguno de los otros tipos de mensajes ICMP lo abarcan, este es un caso extremo ya que indica que el datagrama debe descartarse [2]. El formato para estos mensajes se muestra en la figura 1.21.

0	8	16 31
TIPO (12)	CÓDIGO (0 o 1)	SUMA DE VERIFICACIÓN
INDICADOR	NO UTILIZADO (DEBE SER 0)	
ENCABEZADO DE IP + PRIMEROS 64 BITS DEL DATAGRAMA		
...		

Figura 1.21. Formato de un mensaje ICMP de parámetros incorrectos

En donde:

- INDICADOR, indica el byte del datagrama que causó el problema.

1.4.4.10 Marca de hora

En los sistemas distribuidos, a veces es necesario sincronizar los relojes de las máquinas para que no vaya a existir alguna confusión entre los usuarios del sistema, es entonces cuando una máquina que desea saber la razón del tiempo de otra máquina le envía un mensaje ICMP de solicitud de marca de hora y la máquina destino envía una respuesta de marca de hora a la máquina origen. En la figura 1.22 se muestra el formato de este mensaje [2].

0	8	16 31
TIPO (13 o 14)	CÓDIGO (0)	SUMA DE VERIFICACIÓN
IDENTIFICADOR		NÚMERO DE SECUENCIA
ORIGINAR MARCA DE HORA		
RECIBIR MARCA DE HORA		
TRANSMITIR MARCA DE HORA		

Figura 1.22. Formato de un mensaje ICMP de Marca de hora

En donde:

- IDENTIFICADOR y NÚMERO DE SECUENCIA, permiten que una máquina origen asocie una respuesta con la solicitud correspondiente,
- ORIGINAR MARCA DE HORA, especifica la hora en milisegundos desde la media noche en tiempo universal, y es llenado por la máquina fuente,
- RECIBIR MARCA DE HORA, especifica la hora en que la máquina destino recibió la solicitud, y
- TRANSMITIR MARCA DE HORA, especifica la hora en que se transmitió la respuesta.

1.4.4.11 Solicitud de información y respuesta de información

Los mensajes ICMP de solicitud de información (Tipo 15) y respuesta de información (Tipo 16) están considerados como obsoletos y no deben ser utilizados.

1.4.4.12 Máscara de subred

La máscara de subred es utilizada cuando una máquina necesita identificar en una dirección la red en la que está conectada una máquina y la parte que identifica a la máquina en sí. Para conocer la máscara de subred de la red local, una máquina envía un mensaje de solicitud de máscara de subred a un ruteador y éste envía un mensaje de respuesta de máscara de subred. El formato de estos mensajes [5] se muestra en la figura 1.23.

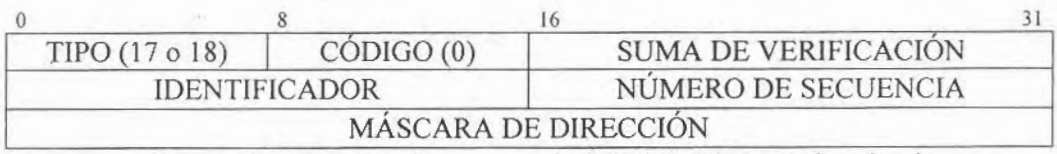


Figura 1.23. Formato de un mensaje ICMP de Máscara de subred

En donde:

- IDENTIFICADOR y NÚMERO DE SECUENCIA, permiten que una máquina origen asocie una respuesta con la solicitud correspondiente, y
- MÁSCARA DE DIRECCIÓN, contiene la máscara de dirección de subred.

Capítulo 2

ATAQUES UTILIZANDO ICMP, ARP Y RARP

En este capítulo se describen los diferentes tipos de ataques que pueden generarse haciendo uso de los protocolos ICMP, ARP y RARP; así como algunas herramientas disponibles que permiten realizar estos ataques y herramientas que analizan el tráfico de un segmento de red.

2.1 TIPOS DE ATAQUES

Cada vez es más el número y la complejidad de los ataques a sistemas computacionales ya sea que estén conectados a una red local o a Internet. Por lo tanto la seguridad es algo que debe importarle a cualquier organización y sobre todo en el caso de que decida conectar su red a Internet, ya que se expone a cualquier ataque que puede poner en peligro la privacidad e integridad de su información y la disponibilidad de sus recursos computacionales.

Los ataques computacionales pueden ser clasificados en dos grupos: ataques pasivos y ataques activos [45]. La diferencia entre los ataques pasivos y los activos, es que en los primeros no se interfiere con el tráfico de la red y en los segundos si, un ejemplo de ataque pasivo es el ataque *sniffing*.

2.1.1 SNIFFING

El *sniffing* es un ataque por medio del cual una máquina captura información que circula por un medio físico, independientemente de que si ésta se encuentra destinada a su dirección física. En este tipo de ataque el intruso no modifica la información sino que simplemente genera un duplicado de ésta para su posterior análisis. La función de *sniffing* puede ser realizada por una gran variedad de dispositivos, siendo el ejemplo más común los analizadores de protocolos. Los ataques basados en *sniffing* comprometen seriamente la seguridad de una red al permitir que un intruso obtenga información de claves de usuarios, números de cuentas bancarias, información de protocolos de bajo nivel y datos privados en general.

Uno de los mayores problemas que se presentan al tratar de enfrentar este tipo de ataque es la facilidad de acceso a herramientas de análisis de protocolos, especialmente si se tiene en cuenta su gran utilidad en procesos de diagnóstico y resolución de problemas. Cualquier persona conectada a una red local puede analizar y capturar el tráfico de su segmento ya sea por medio de un *shareware*[1] o empleando software de bajo costo que corre en cualquier PC. Para hacer esto no hace falta que se trate de un usuario privilegiado de la red (administrador de red), simplemente que se pueda conectar a ella. Una vez obtenidos nombres de usuarios y claves, un intruso tiene una puerta de entrada para acceder al sistema que ha tomado como blanco. Protocolos de capa superior como telnet o ftp son blancos típicos para esta clase de ataque.

En un ataque activo se interfiere con el tráfico legítimo que fluye a través de la red interactuando de manera engañosa con el protocolo de comunicación. A continuación se describen ataques de este tipo.

2.1.2 NEGACIÓN DE SERVICIOS (DoS[2])

En un ataque de negación de servicios el intruso provoca que algún recurso esté demasiado ocupado para responder solicitudes legítimas o para negar el acceso a los usuarios válidos de una determinada máquina, los ataques Teardrop y Land son dos ejemplos de este tipo de ataque [27].

---

[1] El concepto *shareware* implica que el autor cede su programa para ser evaluado por el usuario. Si el usuario decide que el programa es de su interés y que quiere utilizarlo por tiempo indefinido, debe adquirirlo por la cantidad que el autor considere oportuna.

[2] *Denial of Service*.- Negación de Servicios

2.1.2.1 Teardrop

Teardrop consiste en enviar paquetes fragmentados hacía una máquina destino con la opción de reensamblarlo al final, sólo que cuando la máquina intenta llevar a cabo esta tarea se da cuenta que los fragmentos se traslapan entre sí, es decir, hay regiones de datos comunes en dos paquetes distintos provocando que la máquina deje de funcionar.

2.1.2.2 Land

Land consiste en mandar a una máquina y puerto especificado (de la víctima) un paquete con el bit SYN (petición de conexión) activo y con dirección y puerto fuente también de la máquina víctima. Es decir, es como si la máquina víctima quisiera realizar una conexión a ella misma. Esto provoca que la máquina víctima deje de funcionar en la mayoría de los casos o en otros que la carga del CPU alcance valores muy altos.

2.1.3 SPOOFING

El *spoofing* es un ataque en el cual una máquina intrusa se hace pasar por otro equipo de la red, inyectando información adicional en una comunicación. El propósito final de este tipo de ataque consiste en que el intruso pueda hacerse pasar por otro equipo y burlar entonces uno de los principios básicos de la seguridad en redes: la identidad de los participantes de una comunicación.

El mecanismo de *spoofing* a nivel de protocolo ARP e ICMP se describe en las siguientes secciones.

2.2 DESCRIPCIÓN DE ATAQUES ARP

2.2.1 SPOOFING ARP

Los mensajes ARP pueden ser usados para difundir nuevas rutas y robar direcciones IP, una forma de llevarlo a cabo es que en el momento que una máquina origen requiere conocer la dirección física de una máquina destino, la máquina origen envía una solicitud ARP por difusión, logrando con ello que todas las máquinas conectadas a la misma red la reciban, pero sólo la máquina a la que corresponda la dirección IP contenida en el mensaje enviado responda, aunque en el caso de un ataque la máquina llamada intrusa también responderá a dicho mensaje poco tiempo después que la máquina destino, haciendo con ello que la máquina origen tome en cuenta la última respuesta recibida actualizando su tabla de direcciones con la pareja (dirección IP destino real, dirección física intrusa) originando con

ello que la próxima vez que la máquina origen quiera comunicarse con la máquina destino los paquetes sean enviados a la máquina intrusa.

Para completar el ataque ARP la máquina intrusa realiza una solicitud ARP, en la que incluye la dirección IP de la máquina origen y su propia dirección física, con ello la máquina destino actualizará su tabla de direcciones con la pareja (dirección IP origen, dirección física intrusa) ocasionando que cuando la máquina destino le mande información a la máquina origen ésta pase antes por la máquina intrusa.

Así mismo, ARP es utilizado para enviar respuestas falsas aún si una solicitud no ha sido enviada, también se pueden enviar continuamente mensajes ARP buscando con ello mantener los cachés actualizados con información errónea, de esta forma el intruso le crea alias a las interfaces de las máquinas contra las cuales va a realizar un ataque.

Una variación del ataque anterior es usar ARP para hacerse pasar por el *gateway*[3] y filtrar todo el tráfico hacía redes externas. La variación consiste en usar ARP para trazar la ruta del *gateway* IP hacía una dirección física no existente [45].

2.3 *DESCRIPCIÓN DE ATAQUE RARP*

RARP al igual que ARP no provee mecanismos de autentificación, por lo que un intruso puede enviar solicitudes falsas a un servidor real. Para hacer esto el intruso puede asignar la dirección IP de una máquina existente a una estación de trabajo impidiendo el tráfico hacía la máquina víctima.

2.4 *DESCRIPCIÓN DE ATAQUES ICMP*

Debido a los diferentes tipos y funciones de los mensajes ICMP, estos pueden ser utilizados para originar ataques de negación de servicios, a continuación se describen algunos de los más populares [30].

2.4.1 NEGACIÓN DE SERVICIOS EN IRC[4]

El ataque de Negación de Servicios en IRC se lleva a cabo haciendo uso del mensaje ICMP Destino Inaccesible, hay dos formas de implementarlo: Cliente Inaccesible

[3] *Gateway* - Compuerta, Puerta de paso.
[4] *Internet Relay Chat* - Charla o conferencia en Internet

y Servidor Inaccesible. En un ataque Cliente Inaccesible, una máquina remota (la del intruso) envía un paquete Destino Inaccesible a la máquina de un usuario con la dirección falsificada, indicando que proviene del servidor de IRC. Al recibir este paquete la máquina interpretará que el servidor no puede mantener la conexión, por lo que la cerrará. Desde el cliente de IRC sólo se podrá ver que se ha desconectado de IRC. La segunda forma de ataque, Servidor Inaccesible, es lo opuesto a Cliente Inaccesible. Una máquina remota envía un paquete Destino Inaccesible falsificado que indica que proviene de la máquina de un usuario al servidor de IRC. El servidor de IRC termina entonces la conexión. De la misma forma que en el ataque Cliente Inaccesible, sólo se verá que se ha perdido la conexión con el servidor.

2.4.2 SPOOFING ICMP

Los mensajes ICMP Cambio de Ruta, se pueden utilizar para llevar a cabo un ataque de este tipo que consiste en que la máquina intrusa le envía a la máquina origen un mensaje indicándole que debe cambiar la dirección del ruteador a donde envía los datagramas por la dirección de la máquina intrusa. De esta forma se crea un mecanismo de comunicación similar al descrito en SPOOFING ARP.

2.4.3 PING DE LA MUERTE

Consiste en enviar mensajes ICMP Solicitud de Eco más grandes de lo que pueden almacenar las estructuras de datos del núcleo de la víctima (64 Kb), con ello la máquina que recibe estos paquetes en lugar de descartarlos intentará procesarlos, provocando que entre en un ciclo infinito y se quede sin poder responder.

2.4.4 BOMBAS IP O PING FLOODING

Consiste en enviar muchos mensajes ICMP Solicitud de Eco hacía una máquina destino con el fin de que ésta se sature. Este tipo de ataque se puede llevar a cabo no sólo desde una sola máquina sino pueden ser varias a la vez, generando con ello un exceso de tráfico y por lo tanto el consumo excesivo de ancho de banda.

2.4.5 SMURF

Consiste en mandar paquetes ICMP de Solicitud de Eco con una falsa dirección de origen (la IP de la víctima) a una dirección de difusión. De esta forma le llegará a muchas máquinas, cada una de las cuales enviará sus paquetes de respuesta a la dirección origen de la petición (cuyos datos estaban falseados y apuntaban a la víctima). El resultado es que la máquina víctima se ve inundada de paquetes IP, resultando en una saturación de su enlace.

2.5 ESCUCHANDO PAQUETES ICMP QUE CIRCULAN POR LA RED

2.5.1 HERRAMIENTAS PARA ESCUCHAR PAQUETES ICMP

2.5.1.1 Tcpdump

El programa tcpdump solo puede ser ejecutado por el usuario root, y se trata básicamente, de una herramienta de diagnóstico para redes TCP/IP. Puede usarse para analizar el tráfico de la máquina en la que es ejecutado, o el de toda la red local. Para poder escuchar todo el trafico que circula por una red, coloca la interfaz de red en modo promiscuo, que significa que recogerá todos los paquetes de la red, aunque el destinatario no sea su propia dirección física [24].

Ejemplo de la salida de tcpdump:

12:45:31.018620 0:d0:9:f2:75:a3 0:50:fc:27:50:bf 0800 98: 192.168.1.26 > 216.115.109.7: icmp: echo request

4500 0054 002c 0000 4001 7340 c0a8 011a
d873 6d07 0800 9b1e 7f04 0300 3b56 433b
7148 0000 0809 0a0b 0c0d 0e0f 1011 1213
1415 1617 1819

En donde la primer línea indica la hora en la que se originó el mensaje, las direcciones Ethernet origen y destino, las direcciones IP origen y destino e informa que es un mensaje ICMP de tipo solicitud de eco.

La siguiente parte de datos es el datagrama IP y el protocolo encapsulado dentro de él (ICMP), cada uno de los valores corresponden a los campos descritos en el capítulo 1.

2.5.1.2 Exdump

Exdump es una herramienta que captura el tráfico que sale y entra a la máquina en la que es ejecutada, ya que no coloca la interfaz de red en modo promiscuo. Para observar la forma en que presenta los paquetes ICMP Solicitud y Respuesta de eco se puede realizar un ping desde la máquina donde es ejecutada la aplicación [25].

Ejemplo de la salida de exdump:

ICMP Packet Received [Size: 68]
IP Header:
    Header Length: 5

Version: 4
Type of Service: 0
Total Length: 21504
Identification: 11857
Fragment Offset Field: 0
Time to Live: 238
Protocol: 1 (icmp)
Checksum: 15732
Source Address: 216.115.109.7
Destination Address: 192.168.1.26

ICMP Header:
Type: 0 (Echo Reply)
Code: 0 (none)
Checksum: 7199
Echo ID: 1151
Echo Sequence: 2
Gateway: 132223
Fragment MTU: 2

Data:
:VC;÷J

Exdump presenta al usuario el datagrama analizado, es decir, indica el valor que viaja dentro de cada uno de los campos del datagrama IP y del ICMP encapsulado dentro de él.

2.5.1.3 Snort

Snort es un sistema de detección de intrusos, que analiza tráfico y lo coloca en bitácoras en tiempo real. Presenta al final un reporte sobre los paquetes que analizó, tomando en cuenta los paquetes: TCP, UDP, ICMP, ARP, IPv6, IPX, y otros [26].

Ejemplo de la salida de snort:

```
07/04-12:45:32.018644 192.168.1.26 -> 216.115.109.7
ICMP TTL:64 TOS:0x0 ID:45 IpLen:20 DgmLen:84
Type:8  Code:0  ID:32516  Seq:1024  ECHO
3C 56 43 3B 7E 48 00 00 08 09 0A 0B 0C 0D 0E 0F  <VC;~H.....
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F  ................
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F  !"#$%&'()*+,-./
30 31 32 33 34 35 36 37                          01234567
```

Snort presenta la interpretación de los valores para cada uno de los campos de un datagrama IP con un ICMP encapsulado, incluyendo los datos que viajan en el datagrama.

2.5.2 HERRAMIENTAS PARA GENERAR TRÁFICO ICMP SOLICITUD Y RESPUESTA DE ECO

En esta sección se presentan las pruebas realizadas con las herramientas que permiten generar y capturar tráfico ICMP, con el fin de conocer la forma en que trabajan y compararlas con el sistema implementado en este trabajo de tesis.

2.5.2.1 Ping

Ping es un programa que permite verificar si una máquina está accesible por medio de la red, esto lo realiza enviando mensajes ICMP Solicitud de eco, si la máquina cuestionada está accesible ésta le manda a la máquina que realizó el ping mensajes ICMP Respuesta de eco (figura 2.1), con esto se comprueba también que la ruta que siguen los paquetes de la máquina origen a la máquina cuestionada está funcionando adecuadamente.

Paquetes ICMP generados por un ping

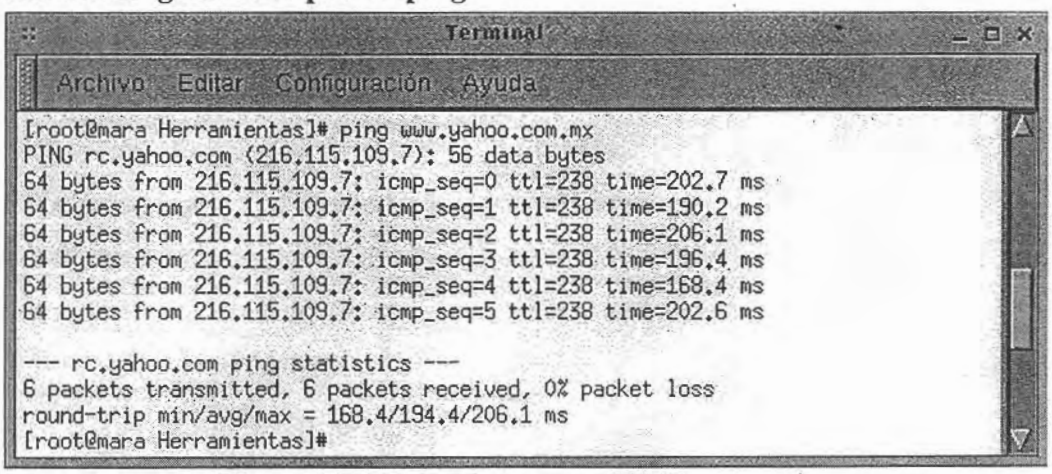


Figura 2.1. Generando paquetes ICMP con un ping

Salida de tcpdump

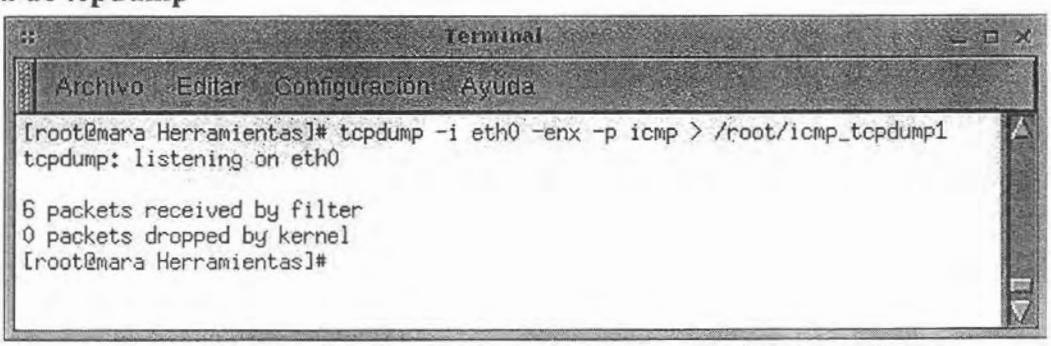


Figura 2.2. Escuchando paquetes generados por el ping con tcpdump

12:45:31.214477 0:50:fc:27:50:bf 0:d0:9:f2:75:a3 0800 98: 216.115.109.7 > 192.168.1.26: icmp: echo reply

4500 0054 5290 0000 ee01 72db d873 6d07
c0a8 011a 0000 a31e 7f04 0300 3b56 433b
7148 0000 0809 0a0b 0c0d 0e0f 1011 1213
1415 1617 1819

12:45:32.018644 0:d0:9:f2:75:a3 0:50:fc:27:50:bf 0800 98: 192.168.1.26 > 216.115.109.7: icmp: echo request

4500 0054 002d 0000 4001 733f c0a8 011a
d873 6d07 0800 8c1e 7f04 0400 3c56 433b
7e48 0000 0809 0a0b 0c0d 0e0f 1011 1213
1415 1617 1819

12:45:32.186313 0:50:fc:27:50:bf 0:d0:9:f2:75:a3 0800 98: 216.115.109.7 > 192.168.1.26: icmp: echo reply

4500 0054 53e0 0000 ee01 718b d873 6d07
c0a8 011a 0000 941e 7f04 0400 3c56 433b
7e48 0000 0809 0a0b 0c0d 0e0f 1011 1213
1415 1617 1819

Salida de snort

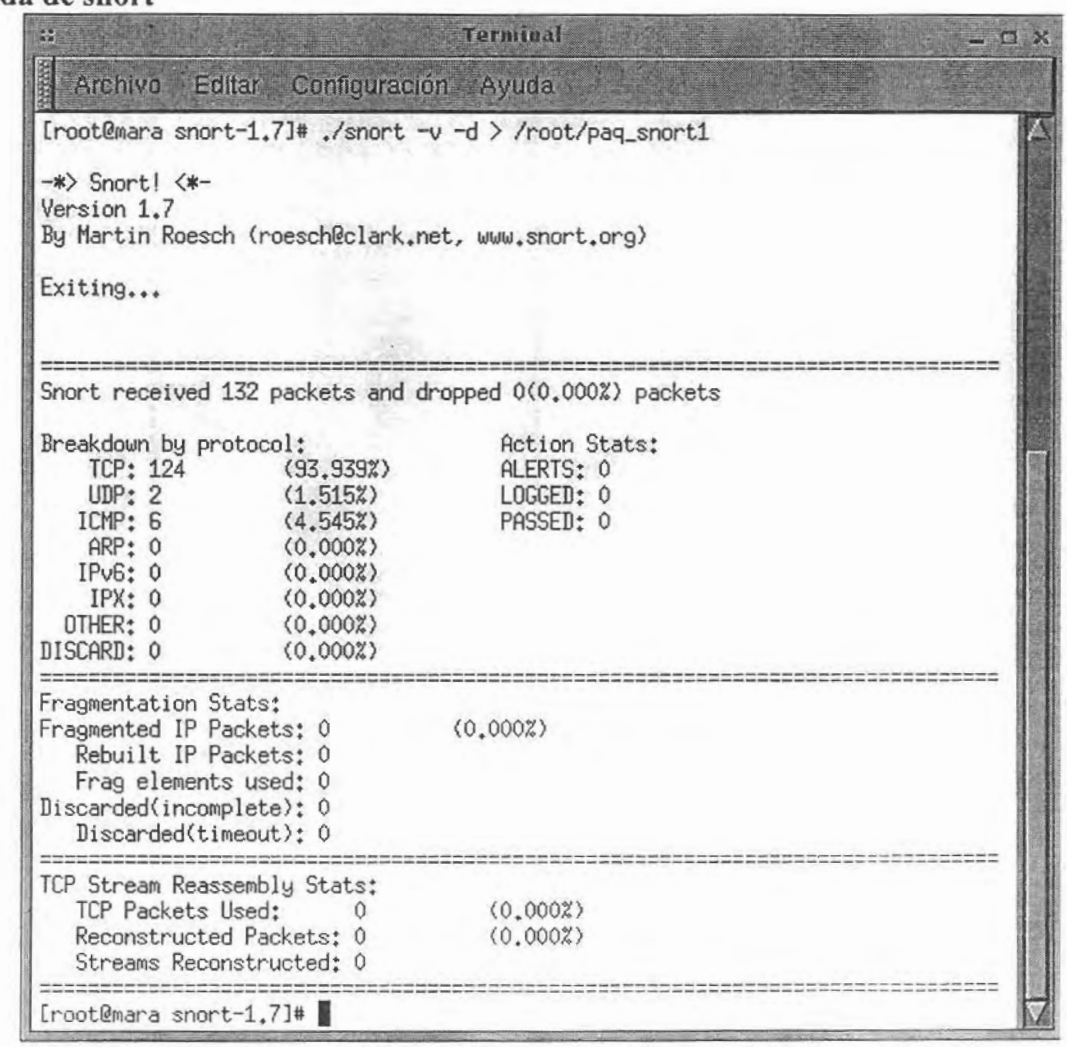


Figura 2.3. Escuchando paquetes generados por el ping con snort

```
07/04-12:45:32.186313 216.115.109.7 -> 192.168.1.26
ICMP TTL:238 TOS:0x0 ID:21472 IpLen:20 DgmLen:84
Type:0  Code:0  ID:32516  Seq:1024  ECHO REPLY
3C 56 43 3B 7E 48 00 00 08 09 0A 0B 0C 0D 0E 0F  <VC;~H.....
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F  ................
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F   !"#$%&'()*+,-./
30 31 32 33 34 35 36 37                          01234567

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

07/04-12:45:32.018644 192.168.1.26 -> 216.115.109.7
ICMP TTL:64 TOS:0x0 ID:45 IpLen:20 DgmLen:84
Type:8  Code:0  ID:32516  Seq:1024  ECHO
3C 56 43 3B 7E 48 00 00 08 09 0A 0B 0C 0D 0E 0F  <VC;~H.....
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F  ................
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F   !"#$%&'()*+,-./
30 31 32 33 34 35 36 37                          01234567

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

07/04-12:45:32.186313 216.115.109.7 -> 192.168.1.26
ICMP TTL:238 TOS:0x0 ID:21472 IpLen:20 DgmLen:84
Type:0  Code:0  ID:32516  Seq:1024  ECHO REPLY
3C 56 43 3B 7E 48 00 00 08 09 0A 0B 0C 0D 0E 0F  <VC;~H.....
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F  ................
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F   !"#$%&'()*+,-./
30 31 32 33 34 35 36 37                          01234567

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Salida de exdump

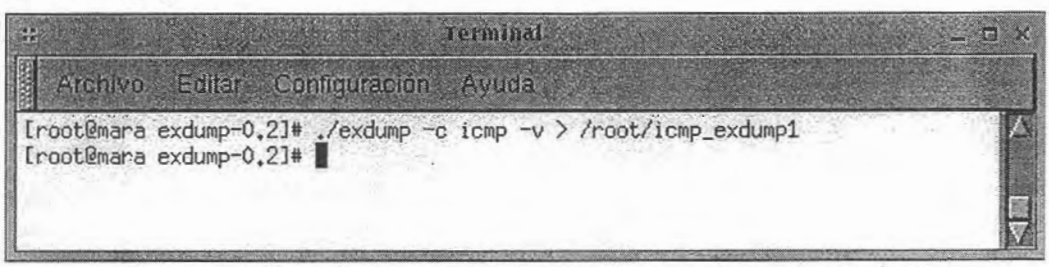


Figura 2.4. Escuchando paquetes generados por el ping con exdump

```
ICMP Packet Received [Size: 68]
IP Header:
        Header Length: 5
        Version: 4
        Type of Service: 0
        Total Length: 21504
        Identification: 36946
        Fragment Offset Field: 0
        Time to Live: 238
        Protocol: 1 (icmp)
```

Checksum: 56178
Source Address: 216.115.109.7
Destination Address: 192.168.1.26

ICMP Header:
Type: 0 (Echo Reply)
Code: 0 (none)
Checksum: 7843
Echo ID: 1151
Echo Sequence: 3
Gateway: 197759
Fragment MTU: 3

Data:
;VC;qH

2.5.2.2 Echok

Echok es un programa que envía mensajes ICMP de Solicitud de eco (figura 2.5). El usuario configura el número de mensajes enviados, el intervalo de tiempo entre el envío de cada paquete, el tamaño del paquete, y la dirección origen (errónea) y destino de ellos [23].

Paquetes ICMP generados por echok

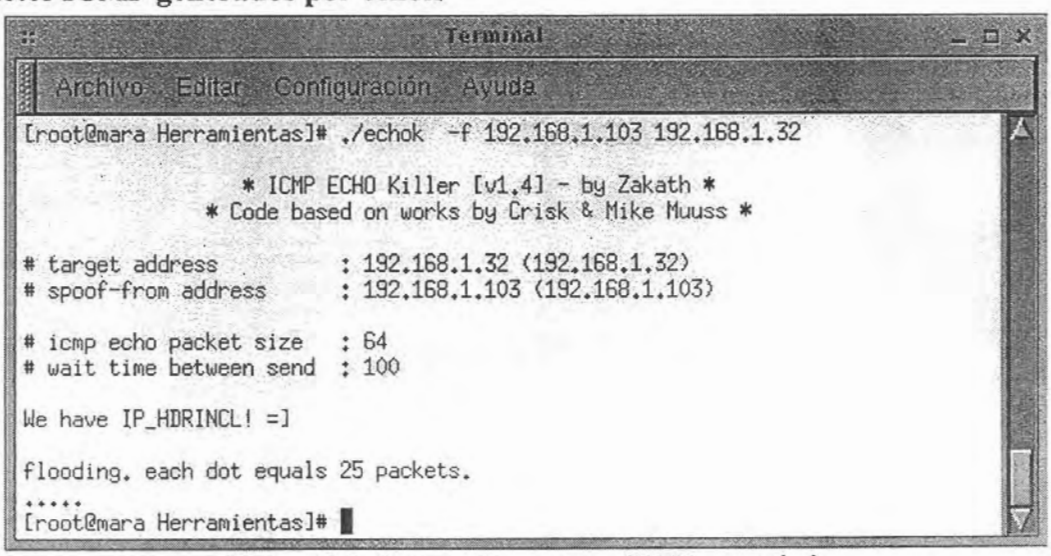


Figura 2.5. Generando paquetes ICMP con echok

Salida de tcpdump

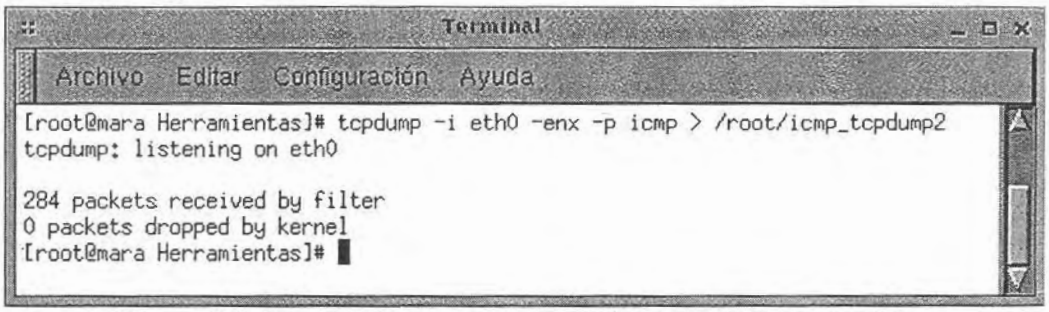


Figura 2.6. Escuchando paquetes generados por echok con tcpdump

13:52:42.863961 0:50:fc:20:89:19 0:50:fc:21:f5:4d 0800 98: 192.168.1.32 > 192.168.1.103: icmp: echo reply

4500 0054 89cc 0000 ff01 ae04 c0a8 0120
c0a8 0167 0000 ffe3 0000 0000 4500 3200
0600 0000 f106 d02f c0a8 0120 0000 0000
0000 0000 0000

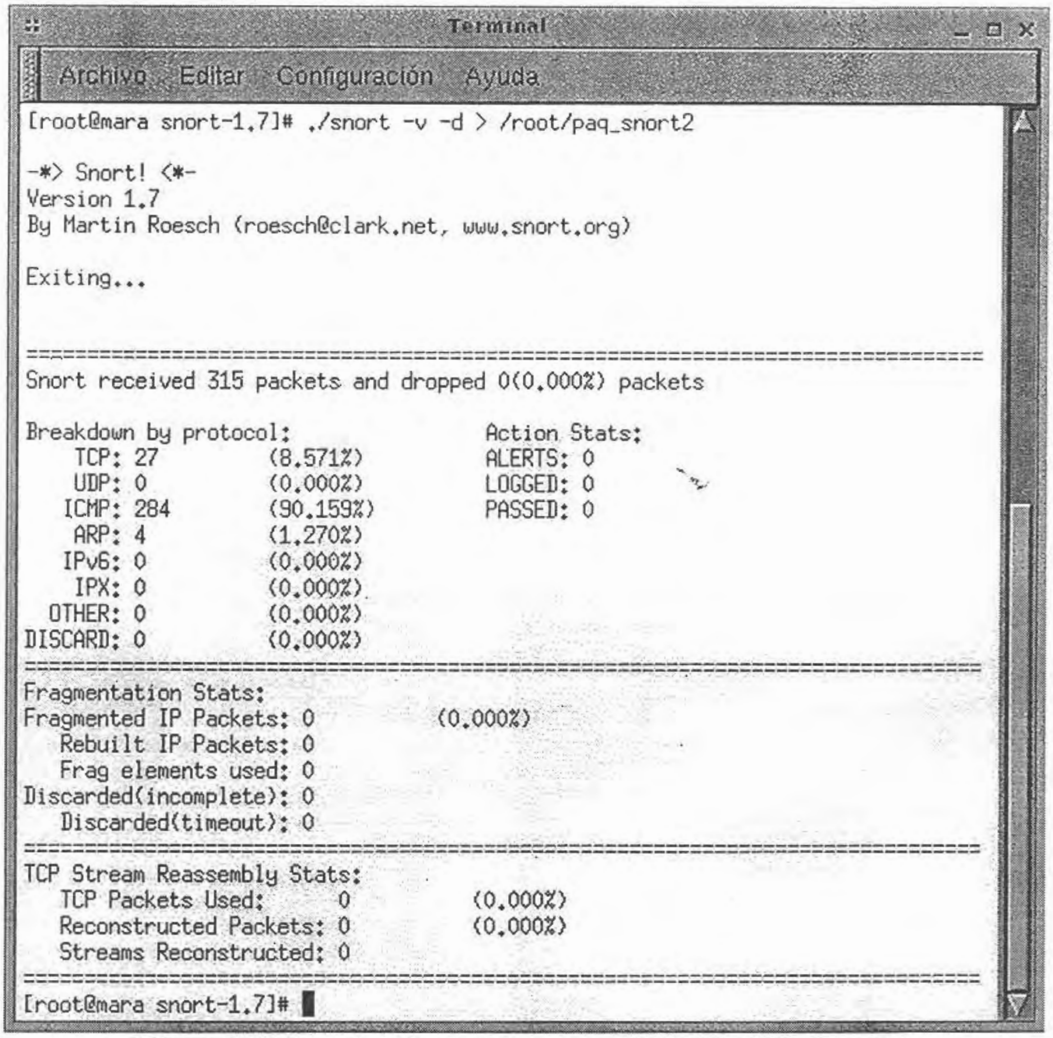
13:52:42.883723 0:d0:9:f2:75:a3 0:50:fc:20:89:19 0800 98: 192.168.1.103 > 192.168.1.32: icmp: echo request

4500 0054 0330 0000 ff01 3ea1 c0a8 0167
c0a8 0120 0800 f7e3 0000 0000 4500 3200
2e00 0000 ed06 ac2f c0a8 0120 0000 0000
0000 0000 0000

13:52:42.883867 0:50:fc:20:89:19 0:50:fc:21:f5:4d 0800 98: 192.168.1.32 > 192.168.1.103: icmp: echo reply

4500 0054 89cd 0000 ff01 ae03 c0a8 0120
c0a8 0167 0000 ffe3 0000 0000 4500 3200
2e00 0000 ed06 ac2f c0a8 0120 0000 0000
0000 0000 0000

Salida de snort




```
[root@mara snort-1.7]# ./snort -v -d > /root/paq_snort2

-*> Snort! <*-
Version 1.7
By Martin Roesch (roesch@clark.net, www.snort.org)

Exiting...


===========================================================================
Snort received 315 packets and dropped 0(0.000%) packets

Breakdown by protocol:                  Action Stats:
    TCP: 27          (8.571%)           ALERTS: 0
    UDP: 0           (0.000%)           LOGGED: 0
   ICMP: 284         (90.159%)          PASSED: 0
    ARP: 4           (1.270%)
   IPv6: 0           (0.000%)
    IPX: 0           (0.000%)
  OTHER: 0           (0.000%)
DISCARD: 0           (0.000%)
===========================================================================
Fragmentation Stats:
Fragmented IP Packets: 0          (0.000%)
   Rebuilt IP Packets: 0
   Frag elements used: 0
Discarded(incomplete): 0
   Discarded(timeout): 0
===========================================================================
TCP Stream Reassembly Stats:
   TCP Packets Used:       0        (0.000%)
   Reconstructed Packets: 0         (0.000%)
   Streams Reconstructed: 0
===========================================================================
[root@mara snort-1.7]#
```

Figura 2.7. Escuchando paquetes generados por echok con snort

```
07/03-13:52:42.863961 192.168.1.32 -> 192.168.1.103
ICMP TTL:255 TOS:0x0 ID:35276 IpLen:20 DgmLen:84
Type:0  Code:0  ID:0  Seq:0  ECHO REPLY
45 00 32 00 06 00 00 00 F1 06 D0 2F C0 A8 01 20  E.2......./...
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 1C 00 00 00 00 00 00                          ........

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

07/03-13:52:42.883723 192.168.1.103 -> 192.168.1.32
ICMP TTL:255 TOS:0x0 ID:816 IpLen:20 DgmLen:84
Type:8  Code:0  ID:0   Seq:0  ECHO
45 00 32 00 2E 00 00 00 ED 06 AC 2F C0 A8 01 20  E.2......./...
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 1C 00 00 00 00 00 00                          ........
```

07/03-13:52:42.883867 192.168.1.32 -> 192.168.1.103
ICMP TTL:255 TOS:0x0 ID:35277 IpLen:20 DgmLen:84
Type:0 Code:0 ID:0 Seq:0 ECHO REPLY
45 00 32 00 2E 00 00 00 ED 06 AC 2F C0 A8 01 20 E.2......../...
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00 1C 00 00 00 00 00 00 ....

=+

2.5.2.3 Smurf

Smurf es un programa que hace que se envíen mensajes ICMP Solicitud de eco desde varias interfaces de red a una sola interfaz de red (figura 2.8), todas especificadas por el usuario [23].

Paquetes ICMP generados por smurf

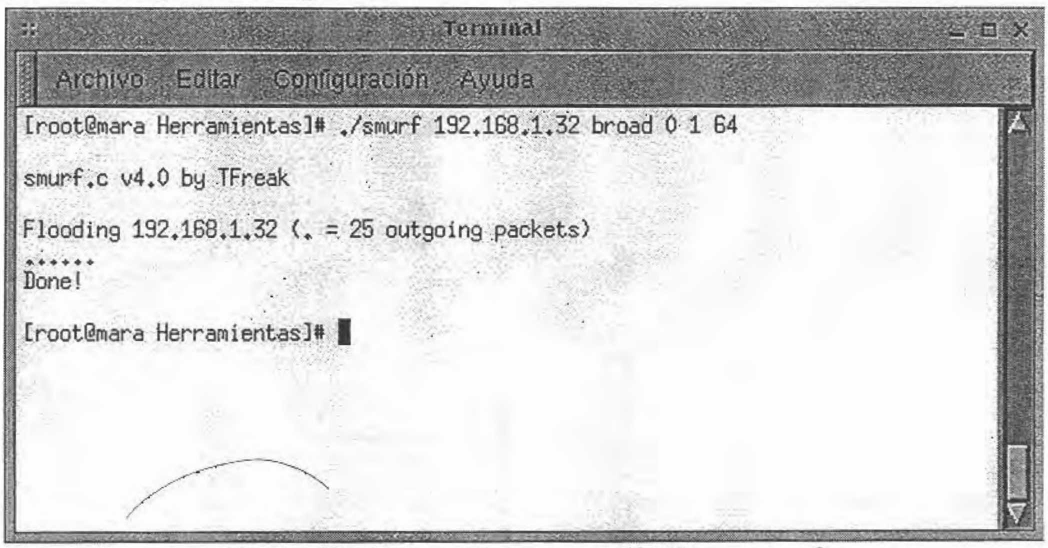


Figura 2.8. Generando paquetes ICMP con smurf

Salida de tcpdump

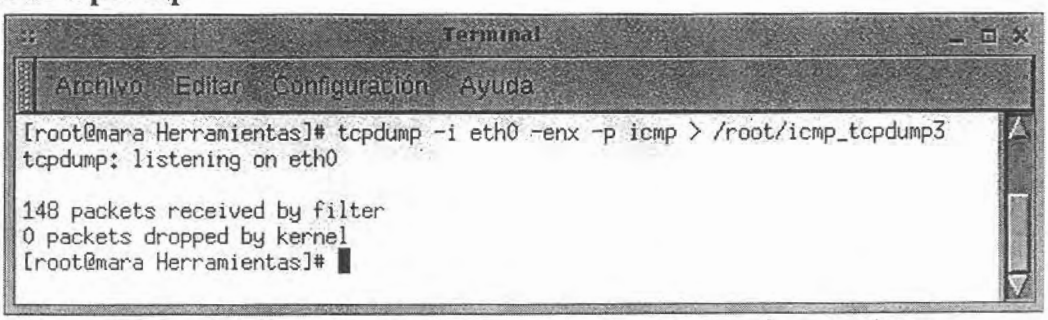


Figura 2.9. Escuchando paquetes generados por smurf con tcpdump

13:58:01.093708 0:d0:9:f2:75:a3 0:50:fc:20:89:19 0800 106: 192.168.1.32 > 192.168.1.24: icmp: echo request

4500 005c 03be 0000 ff01 345a c0a8 0120
c0a8 0118 0800 f7ff 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000

13:58:01.113688 0:d0:9:f2:75:a3 0:50:fc:20:89:19 0800 106: 192.168.1.32 > 192.168.1.25: icmp: echo request

4500 005c 03bf 0000 ff01 3458 c0a8 0120
c0a8 0119 0800 f7ff 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000

13:58:01.133686 0:d0:9:f2:75:a3 0:50:fc:20:89:19 0800 106: 192.168.1.32 > 192.168.1.25: icmp: echo request

4500 005c 03c0 0000 ff01 3457 c0a8 0120
c0a8 0119 0800 f7ff 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000

Salida de snort

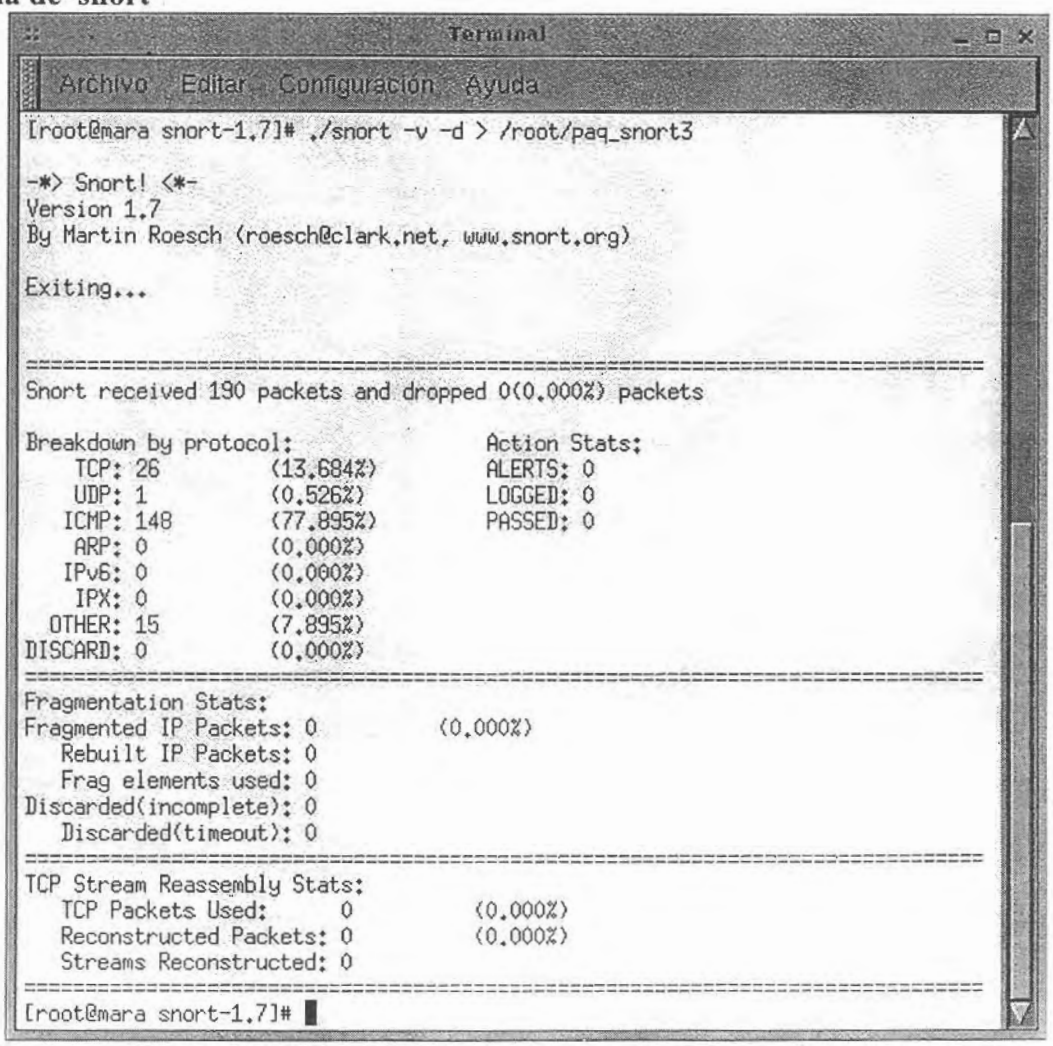


Figura 2.10. Escuchando paquetes generados por smurf con snort

07/03-13:58:01.093708 192.168.1.32 -> 192.168.1.24
ICMP TTL:255 TOS:0x0 ID:958 IpLen:20 DgmLen:92
Type:8 Code:0 ID:0 Seq:0 ECHO
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....

=+

07/03-13:58:01.113688 192.168.1.32 -> 192.168.1.25
ICMP TTL:255 TOS:0x0 ID:959 IpLen:20 DgmLen:92
Type:8 Code:0 ID:0 Seq:0 ECHO
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....

07/03-13:58:01.133686 192.168.1.32 -> 192.168.1.25
ICMP TTL:255 TOS:0x0 ID:960 IpLen:20 DgmLen:92
Type:8 Code:0 ID:0 Seq:0 ECHO
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....

2.5.2.4 Smurf5

Smurf5 es un programa que permite al usuario definir una interfaz de red que va a funcionar como origen de los mensajes generados y definir un archivo broadcast que va a contener las direcciones que van a responder a esos mensajes (figura 2.11). Los tipos de mensajes que puede generar son ICMP, UDP o ambos [23].

Paquetes ICMP generados por smurf5

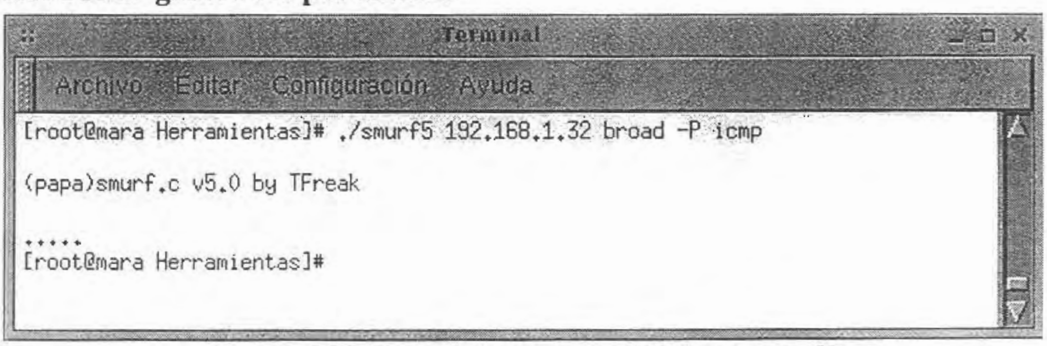


Figura 2.11. Generando paquetes ICMP con smurf5

Salida de tcpdump

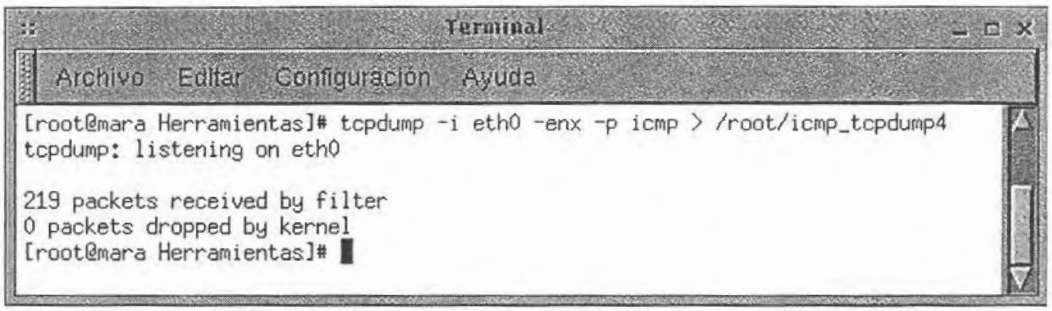


Figura 2.12. Escuchando paquetes generados por smurf5 con tcpdump

14:19:47.354097 0:d0:9:f2:75:a3 0:50:fc:20:89:19 0800 106: 192.168.1.32 > 192.168.1.24: icmp: echo request

4500 005c 0545 0000 ff01 32d3 c0a8 0120
c0a8 0118 0800 f7ff 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000

14:19:47.373699 0:d0:9:f2:75:a3 0:50:fc:20:89:19 0800 106: 192.168.1.32 > 192.168.1.25: icmp: echo request

4500 005c 0545 0000 ff01 32d2 c0a8 0120
c0a8 0119 0800 f7ff 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000

14:19:47.393735 0:d0:9:f2:75:a3 0:50:fc:20:89:19 0800 106: 192.168.1.32 > 192.168.1.23: icmp: echo request

4500 005c 0545 0000 ff01 32d4 c0a8 0120
c0a8 0117 0800 f7ff 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000

Salida de snort

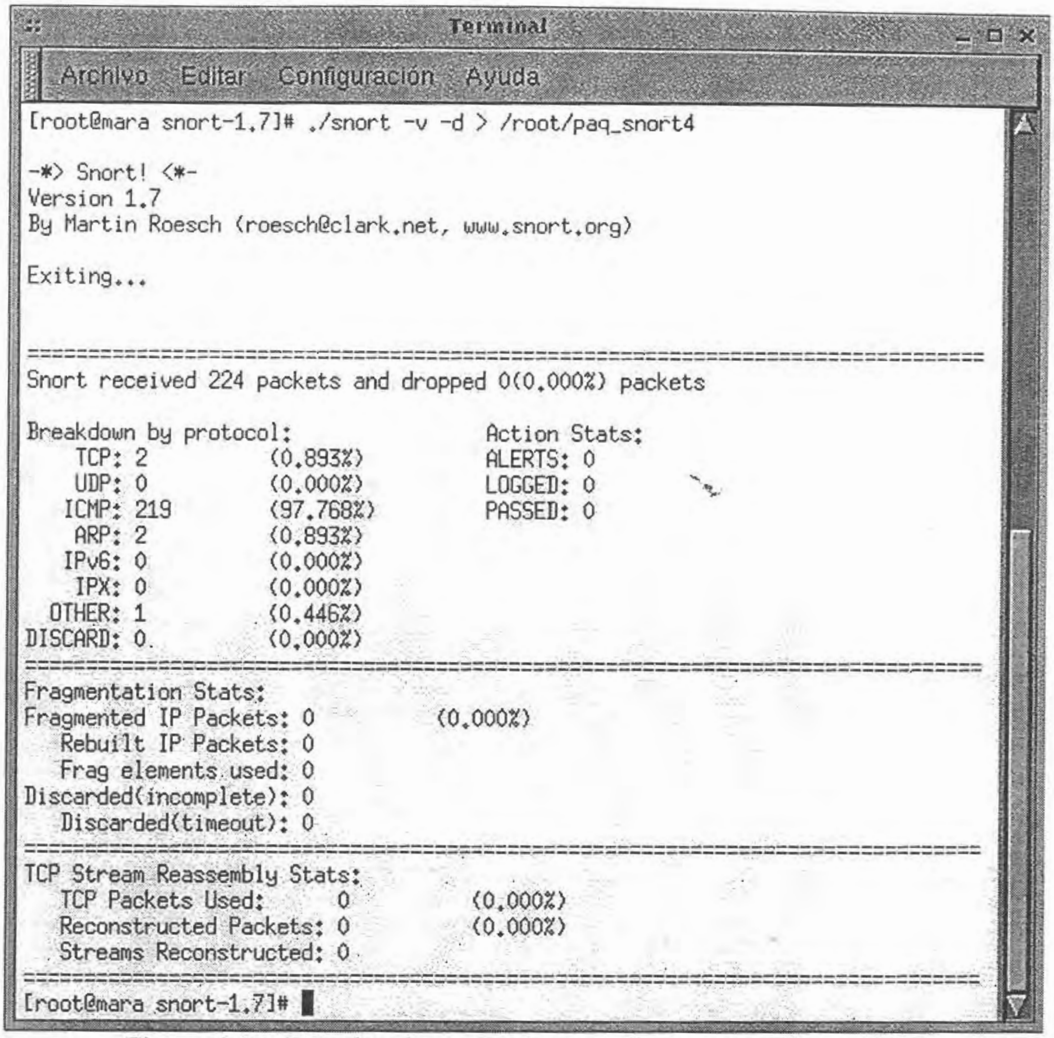


Figura 2.13. Escuchando paquetes generados por smurf5 con snort

```
07/03-14:19:47.354097 192.168.1.32 -> 192.168.1.24
ICMP TTL:255 TOS:0x0 ID:1349 IpLen:20 DgmLen:92
Type:8  Code:0  ID:0   Seq:0  ECHO
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

07/03-14:19:47.373699 192.168.1.32 -> 192.168.1.25
ICMP TTL:255 TOS:0x0 ID:1349 IpLen:20 DgmLen:92
Type:8  Code:0  ID:0   Seq:0  ECHO
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

07/03-14:19:47.393735 192.168.1.32 -> 192.168.1.23
ICMP TTL:255 TOS:0x0 ID:1349 IpLen:20 DgmLen:92
Type:8 Code:0 ID:0 Seq:0 ECHO
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....

2.6 *ESCUCHANDO PAQUETES ARP QUE CIRCULAN POR LA RED.*

Los paquetes ARP capturados son del tráfico normal de la red, ya que no se ha utilizado ninguna herramienta para generarlo. Este tráfico ha sido capturado por medio de tcpdump (figura 2.14) y snort (figura 2.15) que son herramientas que permiten la captura de este tipo de paquetes.

2.6.1 Salida de tcpdump

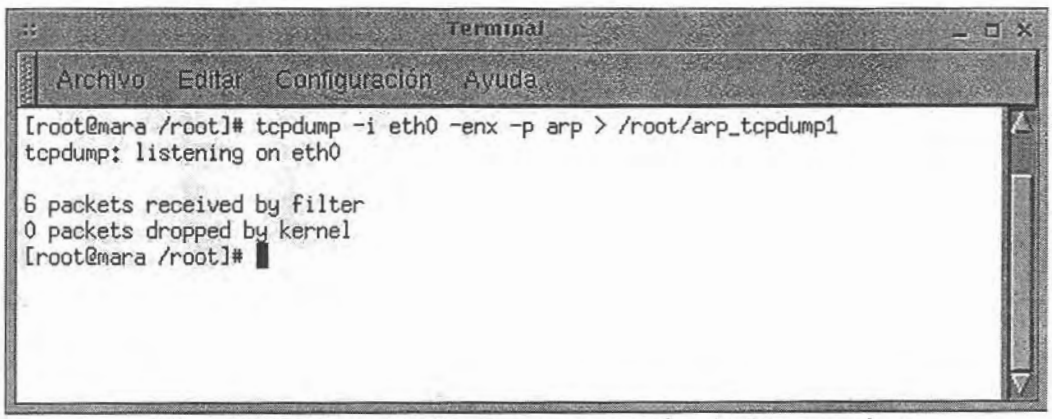


Figura 2.14. Escuchando paquetes generados ARP con tcpdump

21:05:33.597596 0:10:b5:e6:f8:c9 52:54:0:df:d9:d4 0806 60: arp reply 192.168.1.25 is-at 0:10:b5:e6:f8:c9
0001 0800 0604 0002 0010 b5e6 f8c9 c0a8
0119 5254 00df d9d4 c0a8 011f 2020 2020
2020 2020 2020 2020 2020 2020 2020
21:05:36.770664 0:50:fc:20:89:1e ff:ff:ff:ff:ff:ff 0806 60: arp who-has 192.168.1.1 tell 192.168.1.101
0001 0800 0604 0001 0050 fc20 891e c0a8
0165 0000 0000 0000 c0a8 0101 2020 2020
2020 2020 2020 2020 2020 2020 2020
21:05:36.770735 0:50:fc:27:50:bf 0:50:fc:20:89:1e 0806 60: arp reply 192.168.1.1 is-at 0:50:fc:27:50:bf
0001 0800 0604 0002 0050 fc27 50bf c0a8
0101 0050 fc20 891e c0a8 0165 00c4 1e6e
5018 7d78 7d40 0000 4854 5450 2f31

2.6.2 Salida de snort

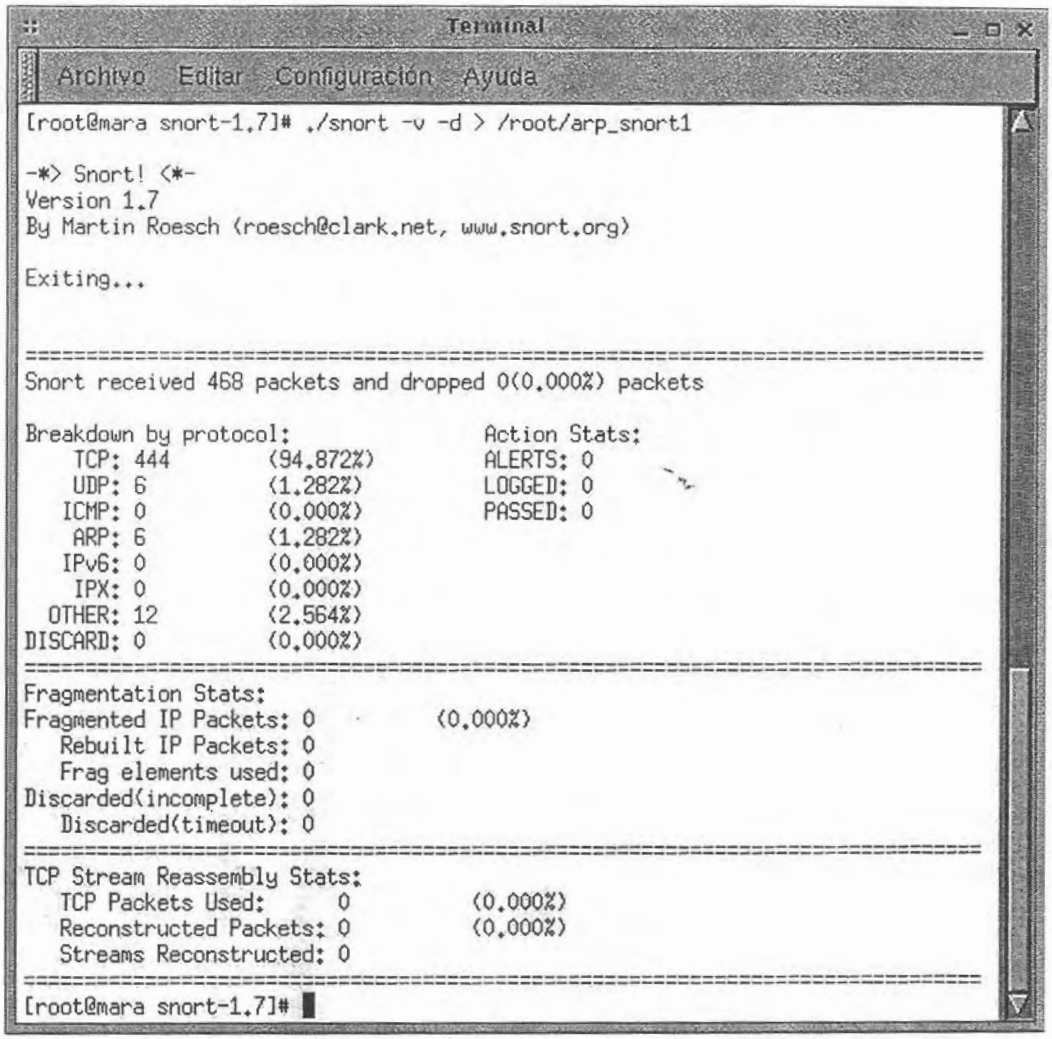


Figura 2.15. Escuchando paquetes generados ARP con snort

Snort no despliega el contenido de los paquetes ARP.

Capítulo 3

IMPLEMENTACIÓN DEL SISTEMA DE MONITOREO

En este capítulo se presenta la implementación del sistema de monitoreo, llamado AVICAR (Analizador y Visualizador de paquetes ICmp, Arp y Rarp), objetivo del desarrollo de esta tesis y motivo por el cual se ha presentado una investigación previa en los capítulos anteriores.

3.1 DISEÑO

El sistema de monitoreo implementado se puede dividir en dos partes:

- La primera de ellas se encarga de escuchar los paquetes que circulan por una LAN[1] y captura sólo a los que transportan mensajes ICMP, ARP y RARP, y
- La segunda parte presenta al usuario los encabezados de los paquetes capturados y la representación grafica del número de paquetes recibidos de cada tipo.

3.1.1 PRIMERA PARTE – Captura de Mensajes

Esta parte está formada por tres módulos, cada uno captura un tipo de paquete diferente (ICMP, ARP, RARP).

[1] *Local Area Network* - Red de Área Local

El algoritmo para cada uno de estos módulos es el siguiente:

1. Definir la estructura en la que va a quedar almacenado el encabezado de cada paquete.
2. Abrir el socket de la familia de Protocolos de Internet para paquetes del tipo de protocolo que se quieran capturar, ya sea ICMP, ARP o RARP.
3. Colocar la tarjeta de red en modo promiscuo.
4. Abrir un archivo de texto en donde se van a ir almacenando los paquetes capturados.
5. Iniciar un ciclo de captura de paquetes.
6. Extraer de la estructura los campos que conforman el encabezado y verificar qué tipo de paquete es el capturado para ir guardándolo en el archivo en el orden deseado con las leyendas necesarias.
7. Terminar con el ciclo cuando se llegue al límite de la condición del mismo.
8. Cerrar el socket.
9. Cerrar el archivo.

3.1.2 SEGUNDA PARTE – Despliegue gráfico de paquetes

Esta parte del sistema de monitoreo está formada por cuatro módulos, un módulo principal por medio del cual se manipulan los tres módulos restantes, los cuales se encargan de ejecutar los programas de la primera parte y de graficar los archivos generados por estos.

El algoritmo para los módulos encargados de graficar es el siguiente:

1. Dibujar las interfaces gráficas.
2. Poner a trabajar el módulo encargado de escuchar paquetes del tipo elegido por el usuario.
3. Abrir el archivo generado en el punto 2.
4. Ir extrayendo línea a línea el contenido del archivo.
5. Analizar cada línea al momento de extraerla para ver de qué tipo de paquete se trata y de esa forma graficarlo en la barra correspondiente dentro de la gráfica temporal, así como presentar la línea extraída en un elemento de texto de la interfaz.
6. Ir formando una gráfica general de los paquetes escuchados.
7. Presentar los porcentajes del número de paquetes recibidos de cada tipo, tanto en la gráfica temporal como en la general.
8. Repetir del paso 1 al 7 hasta que el usuario decida terminar el proceso, guardar el archivo o abrir uno generado con anterioridad.

8.1. Si el usuario decide guardar un archivo
  8.1.1. Extraer los datos del elemento de texto de la interfaz
  8.1.2. Borrar gráficas generadas
  8.1.3. Guardar archivo
8.2. Si el usuario decide abrir un archivo
  8.2.1. Verificar que el archivo seleccionado para abrirse contenga paquetes del tipo de la interfaz gráfica en la que se desea abrir.
  8.2.2. Presentar el contenido del archivo abierto en el elemento de texto de la interfaz.
  8.2.3. Generar gráfica del contenido del archivo.
  8.2.4. Presentar los porcentajes del número de paquetes recibidos de cada tipo.

3.2 DESARROLLO

3.2.1 PRIMERA PARTE – Captura de Mensajes

Los módulos encargados de capturar paquetes se han desarrollado en el lenguaje C, el cual fue inventado e implementado en los años setenta por Dennis Ritchie usando UNIX como sistema operativo, es un lenguaje de nivel medio ya que combina elementos de lenguajes de alto nivel con la funcionalidad del lenguaje ensamblador. Este lenguaje provee las estructuras necesarias para crear los módulos de captura de paquetes, dichas estructuras se detallan en la siguiente sección.

3.2.1.1 Estructuras definidas para encabezados TCP/IP

El lenguaje C incluye librerías en las que están declaradas estructuras de datos que permiten el almacenamiento del encabezado de paquetes de un tipo específico. Las estructuras utilizadas en el desarrollo de esta herramienta de monitoreo son:

- **struct ethhdr,** declarada en <linux/if_ether.h> y define el formato del encabezado Ethernet.

```
struct ethhdr {
    unsigned char  h_dest[ETH_ALEN];      /* dirección eth destino */
    unsigned char  h_source[ETH_ALEN];    /* dirección eth origen  */
    unsigned short h_proto;               /* campo ID de tipo de paquete */
};
```

- **struct iphdr,** la cual está declarada en <netinet/ip.h> y define el formato del encabezado IP.

```
struct iphdr {
    #if __BYTE_ORDER == __LITTLE_ENDIAN
      unsigned int ihl:4;
      unsigned int version:4;
    #elif __BYTE_ORDER == __BIG_ENDIAN
      unsigned int version:4;
      unsigned int ihl:4;
    #else
    # error  "Please fix <bits/endian.h>"
    #endif
      u_int8_t tos;
      u_int16_t tot_len;
      u_int16_t id;
      u_int16_t frag_off;
      u_int8_t ttl;
      u_int8_t protocol;
      u_int16_t check;
      u_int32_t saddr;
      u_int32_t daddr;
      /*Las opciones empiezan aquí. */
    };
```

- **struct icmphdr**, esta estructura está declarada en <netinet/ip_icmp.h> y define el formato del encabezado ICMP.

```
struct icmphdr
{
 u_int8_t type;          /* tipo de mensaje */
 u_int8_t code;          /* tipo de sub-código*/
 u_int16_t checksum;
 union
 {
  struct
  {
   u_int16_t   id;
   u_int16_t   sequence;
  } echo;                          /* eco datagrama */
  u_int32_t    gateway;           /* dirección gateway*/
  struct
  {
   u_int16_t   __unused;
   u_int16_t   mtu;
  } frag;                          /* descubrimiento de mtu*/
 } un;
};
```

- **struct arphdr,** está declarada en <net/if_arp.h> y define el encabezado de ARP o RARP, esto último depende del valor del campo ar_op, ya que toma el valor de 1 o 2

para los casos de Solicitud y Respuesta ARP, respectivamente, y el valor de 3 o 4 para Solicitud o Respuesta RARP, según sea el caso.

```
struct arphdr
 {
      unsigned short int ar_hrd;    /* Formato de la dir. de hardware.  */
      unsigned short int ar_pro;    /* Formato de la dir. de protocolo. */
      unsigned char ar_hln;              /* Longitud de la dir. de hardware. */
      unsigned char ar_pln;              /* Longitud de la dir. de protocolo.*/
      unsigned short int ar_op;          /* Código ARP. */
    #if 0
      /* Ethernet es parecido a esto: Esta parte es de tamaño variable,
           sin embargo... */
      unsigned char __ar_sha[ETH_ALEN];      /* Dir. de hw de transmisor.  */
      unsigned char __ar_sip[4];             /* Dir. IP del transmisor.  */
      unsigned char __ar_tha[ETH_ALEN];      /* Dir. de hw del objetivo.  */
      unsigned char __ar_tip[4];             /* Dir. IP del objetivo.  */
    #endif
 };
```

- **struct ether_arp,** está declarada en <netinet/if_ether.h> y define el encabezado de Ethernet que incluye a un ARP o RARP

```
struct  ether_arp
{
    struct   arphdr ea_hdr;          /* encabezado de tamaño fijo */
    u_int8_t arp_sha[ETH_ALEN];/* Dir. de hw de transmisor  */
    u_int8_t arp_spa[4];             /* Dir. de protocolo del transmisor */
    u_int8_t arp_tha[ETH_ALEN]; /* Dir. de hw del objetivo */
    u_int8_t arp_tpa[4];             /* Dir. de protocolo del objetivo */
};
```

Hasta aquí se han presentado las estructuras de datos que vienen definidas por el lenguaje, las siguientes son estructuras que se definieron para la implementación del sistema de monitoreo.

```
/* estructura definida para guardar un datagrama ICMP que viaja por Ethernet */

struct etherpacket {
     struct ethhdr      eth;
     struct iphdr       ip;
     struct icmphdr     icmp;
     };

/* estructura definida para guardar un datagrama ARP que viaja por Ethernet */
struct etherpacket {
     struct ethhdr      eth;
```

```
    struct ether_arp    arp;
    };

/* estructura definida para guardar un datagrama RARP que viaja por Ethernet */

struct etherpacket {
    struct ethhdr       eth;
    struct ether_arp    rarp;
    };
```

Estas dos últimas estructuras se pueden fusionar en una sola, ya que lo que varía no son las estructuras en sí, sino el valor de uno de los campos dentro de ellas.

Para poder realizar la captura de paquetes, es necesario establecer una comunicación con la interfaz de red, por lo que en seguida se explica la forma en que se realiza esta comunicación.

3.2.1.2 Sockets

Los sockets son mecanismos que permiten la comunicación entre procesos, aún cuando estos procesos se estén ejecutando en diferentes máquinas. La llamada para abrir uno de estos canales de comunicación es *socket* [39], éste devuelve un descriptor de fichero válido en caso de no haber ningún error, de lo contrario devuelve –1. La forma en que se declara es:

```
#include <sys/types.h>
#include <sys/socket.h>

/* Para el caso de este sistema de monitoreo que está implementado en Linux */
#include <linux/socket.h>

int socket(familia,tipo,protocolo)
int familia, tipo, protocolo;
```

En donde:

- familia, especifica la familia de *sockets* o familia de direcciones que se desea emplear. Los diferentes tipos de familias están definidos en <sys/socket.h> y dependen del sistema o de la configuración del hardware. Las familias que regularmente están presentes en todos los sistemas son:

AF_UNIX	Es la familia de *sockets* que se utiliza para la comunicación entre procesos que se ejecutan en la misma máquina.
AF_INET	Es la familia de *sockets* que se utiliza para la comunicación entre procesos que son ejecutados en diferentes máquinas, esta comunicación se realiza por medio de protocolos, como TCP o UDP.

Existen otras familias como:

AF_CCITT	Norma X.25 del CCITT
AF_NS	Protocolos NS de Xerox.

- tipo, indica la semántica de la comunicación para el *socket* y puede ser:

SOCK_STREAM	Provee un flujo de datos bidireccional, secuenciado, sin duplicación de paquetes y libre de errores, utiliza el protocolo TCP.
SOCK_SEQPACKET	Tiene las características de SOCK_STREAM pero con la diferencia de que el tamaño de los mensajes es fijo.
SOCK_DGRAM	Provee un flujo de datos bidireccional, donde los paquetes pueden llegar fuera de secuencia, no llegar o contener errores, utiliza el protocolo UDP.
SOCK_RAW	Permite el acceso a protocolos de nivel más bajo como el IP o a las interfaces de red.
SOCK_PACKET	Es similar a SOCK_RAW, aunque para los sistemas LINUX kernel 2.0 sólo es soportado como SOCK_PACKET.

- protocolo, especifica el protocolo que se va a utilizar en el *socket*. Puede valer 0 si la elección del protocolo se deja en manos del sistema.

Dentro de los módulos construidos en el desarrollo del sistema de monitoreo, los *sockets* se han declarado de la siguiente forma:

```
if_fd=socket(AF_INET,SOCK_PACKET,PROTO)
```

donde, PROTO, toma el valor del protocolo del cual se quieren escuchar paquetes, la manera en que se le asignó un valor es:

```
#define PROTO htons(0x0800)  /* Codigo Ethernet para protocolo IP */
#define PROTO htons(0x0806)  /* Codigo Ethernet para protocolo ARP */
#define PROTO htons(0x8035)  /* Codigo Ethernet para protocolo RARP */
```

Una vez que se ha abierto el canal de comunicación entre procesos, es necesario leer los datos del socket, la forma en que se declara la función que se utiliza para esto es:

```
int recvfrom (descriptor, buf, long, banderas, dest, destl)
int descriptor;
void *buf;
int long, banderas;
void *det;
int destl;
```

En donde:

- descriptor, es el descriptor que devuelve el *socket*.
- buffer, es un puntero al buffer donde se van a escribir los datos leídos.
- longitud, es el número de máximo de bytes que se pueden escribir en el buffer.
- dest, apunta a una estructura en la cual se almacena la dirección del socket origen de los datos.
- destl, es un parámetro de entrada/salida, al llamar a la función recvfrom, contiene el tamaño de la estructura apuntada por dest, y al salir de la función, tiene el valor real de la dirección leída.

Para *sockets* basados en paso de mensajes, cada mensaje se debe leer completo en una sola operación, si el mensaje es demasiado grande, entonces se trunca.

La forma en que se leen los datos del *socket* en la herramienta de monitoreo implementada es:

```
recvfrom(if_fd,&ep,sizeof(ep),0,&dest,&dlen);
```

En donde:

- ep, es el nombre de la estructura definida para recibir los mensajes de los sockets, que en este caso contendrá los encabezados de los paquetes que se quieran escuchar.

3.2.1.3 Habilitación del modo promiscuo de la interfaz de red

Las interfaces de red tienen distintos modos de funcionamiento, de recepción y transmisión de paquetes, uno de estos modos es el llamado modo promiscuo, el cual permite a la interfaz recibir y procesar todos los paquetes que circular por su red local, en lugar de hacerlo sólo para los paquetes que van destinados a su dirección física y a transmisiones broadcast, que es su modo de funcionamiento estándar.

En sistemas UNIX, la interfaz de red se coloca en modo promiscuo obteniendo y estableciendo las banderas de la interfaz mediante una llamada al sistema *ioctl*.

Se realiza la operación *ioctl* sobre el *socket* para obtener la configuración de las interfaces de red del sistema [39], su declaración es:

```
#include <sys/ioctl.h>
int ioctl(descriptor, operacion, arg)
int descriptor, operación;
```

En donde:

- descriptor, es el descriptor que devuelve el socket.
- operacion, se refiere al tipo de operación que se va a realizar y los valores que puede tomar dependen del tipo de dispositivo con el que se esté trabajando.
- arg, contiene los parámetros con los que se va a programar un dispositivo determinado.

La forma en que se aplica esta operación en los módulos de captura de paquetes del sistema de monitoreo es:

```
ioctl(if_fd, SIOCGIFFLAGS, &ifr)
```

Donde SIOCGIFFLAGS, es la operación para obtener las banderas de un dispositivo y el tercer parámetro es un puntero a una *struct ifreq* que está definida en <linux/if.h>, dicha estructura contiene el nombre de la interfaz de red en su campo *ifr_name*. Así, después de ejecutar esta operación, la estructura *ifreq* contiene en su campo *ifr_flags* las banderas de la interfaz de red de la que se utilizó su nombre como parámetro.

Después de obtener las banderas, es necesario modificarlas para colocar la interfaz de red en modo promiscuo, para esto se requiere efectuar una operación lógica OR entre las banderas originales y la constante IFF_PROMISC como lo muestra la siguiente línea de código:

```
ifr.ifr_flags |= IFF_PROMISC;
```

Una vez que se ha realizado esta operación lógica, las banderas pueden ser modificadas haciendo un llamado a *ioctl* con la operación SIOCSIFFLAGS, dicha operación configura las banderas de la interfaz de red y se realiza de la siguiente forma:

```
ioctl(if_fd2, SIOCSIFFLAGS, &ifr)
```

Hasta este punto se han descrito los puntos importantes para construir los módulos encargados de capturar los paquetes que interesan en el desarrollo de esta herramienta de monitoreo.

3.2.2 SEGUNDA PARTE – Despliegue gráfico de paquetes

Los módulos que se encargan de graficar los paquetes capturados se han implementado con Tcl/Tk. Tcl fue creado por John K. Ousterhout y su equipo de la Universidad de California, es un lenguaje de programación interpretado que corre bajo Windows, Windows NT, UNIX, y Mac Os. Tcl es un lenguaje de comandos que pueden incrementarse al implementarlos por medio de funciones en C/C++, así mismo, es posible embeber aplicaciones en C/C++ dentro del intérprete de Tcl, lo cual origina nuevas versiones de Tcl, llamadas extensiones. Una de las extensiones más conocida y que es distribuida con Tcl, es Tk, creada también por Ousterhout. Esta extensión añade a los comandos de Tcl comandos que permiten crear interfaces gráficas, tales como ventanas, botones, menús, barras de scroll, y varios elementos más, que se conocen como widgets.

Tcl proporciona dos intérpretes de comandos, para Tcl, tclsh80 y para Tk, wish80, por lo que en la implementación de la interfaz gráfica del sistema de monitoreo se ha utilizado el intérprete de Tk. Tcl proporciona la facilidad para ejecutar código realizado en C y para el manejo de listas y Tk rapidez para construir interfaces gráficas [38].

La implementación de las interfaz gráfica se ha realizado por medio de scripts, estos contienen como primer línea de código una llamada al intérprete utilizado, de la forma:

```
#!/usr/bin/wish
```

Esta segunda parte del sistema de monitoreo está formada por un script principal y tres scripts independientes, el primero dibuja el menú por medio del cual se pueden activar los siguientes tres scripts independientes, cuya función es desplegar y graficar la información de los paquetes ICMP, ARP y RARP.

La forma en que el script principal ejecuta los tres scripts independientes es:

```
exec icmp.tcl &
exec arp.tcl &
exec rarp.tcl &
```

exec es usado para ejecutar otros programas desde una aplicación Tcl, el segundo parámetro es el nombre del programa a ejecutar y el símbolo "&", significa que el comando se ejecutará en segundo plano.

De la misma forma, desde cada uno de estos scripts se ejecutan las aplicaciones realizadas en C, que son las que capturan los paquetes.

Para el registro de los diferentes tipos de paquetes escuchados se ha hecho uso de listas. Una lista en Tcl es un conjunto de elementos, tales como 1 2 3 4 5 seis, deben encerrarse entre comillas o llaves para agruparlos. Tanto los elementos como la lista en sí son considerados tipo string, como todas las cosas en Tcl. Existen varios comandos que facilitan la manipulación de las listas, dichos comandos se presentan en la tabla 3.1, y algunos de ellos se han utilizado en el desarrollo de la interfaz gráfica.

Tabla 3.1. Comandos de manejo de listas

COMANDO	DESCRIPCIÓN
list arg1 arg2	Crea una lista con los argumentos
lindex lista i	Devuelve el elemento i-ésimo de la lista
llength lista i j	Devuelve la longitud de la lista
lreplace lista i j arg arg	Reemplaza los elementos desde el i al j por los argumentos. Si no hay argumentos y i=j, borra este elemento. Devuelve una nueva lista.

En el desarrollo de la interfaz gráfica se ha hecho uso de elementos, tales como: botones, etiquetas, frames, widgets de texto, canvas, entre otros. En la tabla 3.2 se muestran los comandos de creación de widgets, y las opciones comunes para cada uno de ellos se muestran en la tabla 3.3.

Tabla 3.2. Comandos de creación de widgets

WIDGET	DESCRIPCIÓN
button	Crea un botón que ejecuta un comando
canvas	Crea un área donde se puede dibujar.
checkbutton	Crea un botón de chequeo. Tiene dos posiciones, activado o desactivado.
entry	Crea una entrada de texto. Se utiliza para introducir datos.
frame	Crea un contenedor de widgets.
label	Crea una etiqueta. Sólo es de lectura. No se puede escribir en ella.
listbox	Crea una lista de texto.
menu	Crea un menú.
menubutton	Crea un botón que presenta un menú. Serían los botones existentes en la barra de menú.
message	Crea un cuadro de mensaje, sólo de lectura.
radiobutton	Crea un botón de radio. Puede estar activado sólo uno a la vez.
scale	Crea una escala que permite ajustar el valor de una variable con su desplazamiento.
scrollbar	Crea las barras de scroll, tanto verticales como horizontales.
text	Crea texto editable o no, de propósito general.
toplevel	Crea una ventana.

Tabla 3.3. Opciones comunes para widgets

OPCIÓN	DESCRIPCIÓN
-activebackground color	Color de fondo cuando está activo.
-activeborderwidth width	Ancho del borde, en pixeles, cuando está activo.
-activeforeground color	Color de primer plano cuando está activo.
-anchor anchor_pos	Información de posiciones dentro del widget, al n, ne, nw, s, se, sw, e, w, y center.
-background color	Configura a un widget normal el color de fondo.
-borderwidth width	Configura el ancho del borde, en pixeles.
-command script_tcl	Ejecuta script_tcl cuando es invocado.
-font nombre_fuente	Usa el nombre_fuente para el texto de los widgets.
-geometry anchoxalto	Configura el tamaño del widget, normalmente en pixeles, pero se pueden usar diferentes unidades. La x es requerida.
-jump on_o_off	Si es verdadero, los scrollbar retrasan la actualización hasta que el botón del mouse es soltado.
-orient orientación	Configura la orientación a horizontal o vertical.
-padx pad	Rellena pixeles extras en la dirección X.
-pady pad	Rellena pixeles extras en la dirección Y.
-relief relieve	Coloca un relieve en 3D del tipo flat, groove, raised, ridge, solid, o sunken.

Tabla 3.3. Opciones comunes para widgets

OPCIÓN	DESCRIPCIÓN
-text string	Configura el texto a desplegar.
-textvariable nombre_var	Configura la variable a usar para obtener el texto a desplegar.
-width ancho	Configura el ancho, normalmente en pixeles, pero se pueden usar diferentes unidades.
-xscrollcommand prefijo	Prefijo para comando usado para la comunicación con scrollbars horizontales.
-yscrollcommand prefijo	Prefijo para comando usado para la comunicación con scrollbars verticales.

En la tabla 3.4 se muestran los comandos de manipulación de widgets.

Tabla 3.4. Comandos de manipulación de widgets

COMANDO	DESCRIPCIÓN
after	Ejecuta un comando después de un cierto periodo de tiempo.
bell	Hace sonar un pitido.
bind	Asocia un comando Tcl con un evento del sistema.
clipboard	Manipula el portapapeles.
destroy	Destruye un widget.
fileevent	Asocia un comando Tcl con un descriptor de fichero.
focus	Controla el foco de entrada.
image	Crea y manipula imágenes.
option	Accede a la base de datos de los recursos.
pack	Administrador de geometría.
place	Administrador de geometría.
selection	Manipula la selección.
send	Envia un comando Tcl a otro programa Tcl.
tk	Informa sobre el estado interno de Tk.
tkerror	Captura los errores.
tkwait	Detiene el programa en espera de un evento.
update	Actualiza la pantalla.
winfo	Informa acerca de las ventanas.
wm	Interactúa con el administrador de ventana.

3.3 RESULTADOS

Una vez implementados los módulos que componen el Sistema de Monitoreo, se obtiene la herramienta completa llamada AVICAR, esta herramienta permite al usuario (administrador de red) capturar y visualizar los paquetes ICMP, ARP y RARP, así como las gráficas del número de paquetes recibidos de cada tipo. Se pueden escuchar los tres tipos de paquetes a un mismo tiempo, hasta que el usuario decida detener alguna o todas las aplicaciones, cada una de ellas tiene la opción de Guardar los paquetes capturados, así como de Abrir archivos guardados anteriormente, generando la gráfica a partir del contenido del archivo elegido.

Los módulos que escuchan paquetes pueden considerarse sniffers, aunque su tarea se reduce a capturar los paquetes y respaldar únicamente los encabezados de ellos, no los datos que transmiten. De esta forma su utilidad se encuentra en que a partir de estos módulos es posible analizar los paquetes para saber hacía qué equipo hay mayor tráfico de información, así como saber la cantidad de paquetes que circulan en mayor o menor grado por una red determinada.

La interfaz gráfica que presenta este sistema de monitoreo es sencilla y de fácil utilización, permitiendo al usuario no perder tiempo en el aprendizaje de su uso.

Una de las desventajas que presenta este sistema, es que el tiempo que se tarda en graficar el contenido de los archivos generados por los módulos que capturan los paquetes, deja de escuchar el tráfico y por lo tanto hay pérdida de paquetes. Con ello, este sistema permite analizar sólo muestras del tráfico de la red, no de todo el tráfico.

Después de haber descrito el funcionamiento del sistema de monitoreo, se presentan las pantallas de AVICAR.

La pantalla principal de AVICAR se muestra en la figura 3.1.

Figura 3.1. Pantalla principal de AVICAR

Al activar la opción de CAPTURAR PAQUETES ICMP, el menú se presenta como lo muestra la figura 3.2. La pantalla donde se visualizan los paquetes escuchados, así como la gráfica que representa el número de paquetes recibidos por intervalos de tiempo se

muestra en la figura 3.3. Si el usuario desea ver una gráfica general de los paquetes recibidos en un tiempo determinado, es posible visualizarla. La manera en que se presenta es una ventana independiente y se muestra en la figura 3.4.

Figura 3.2. Capturar paquetes ICMP

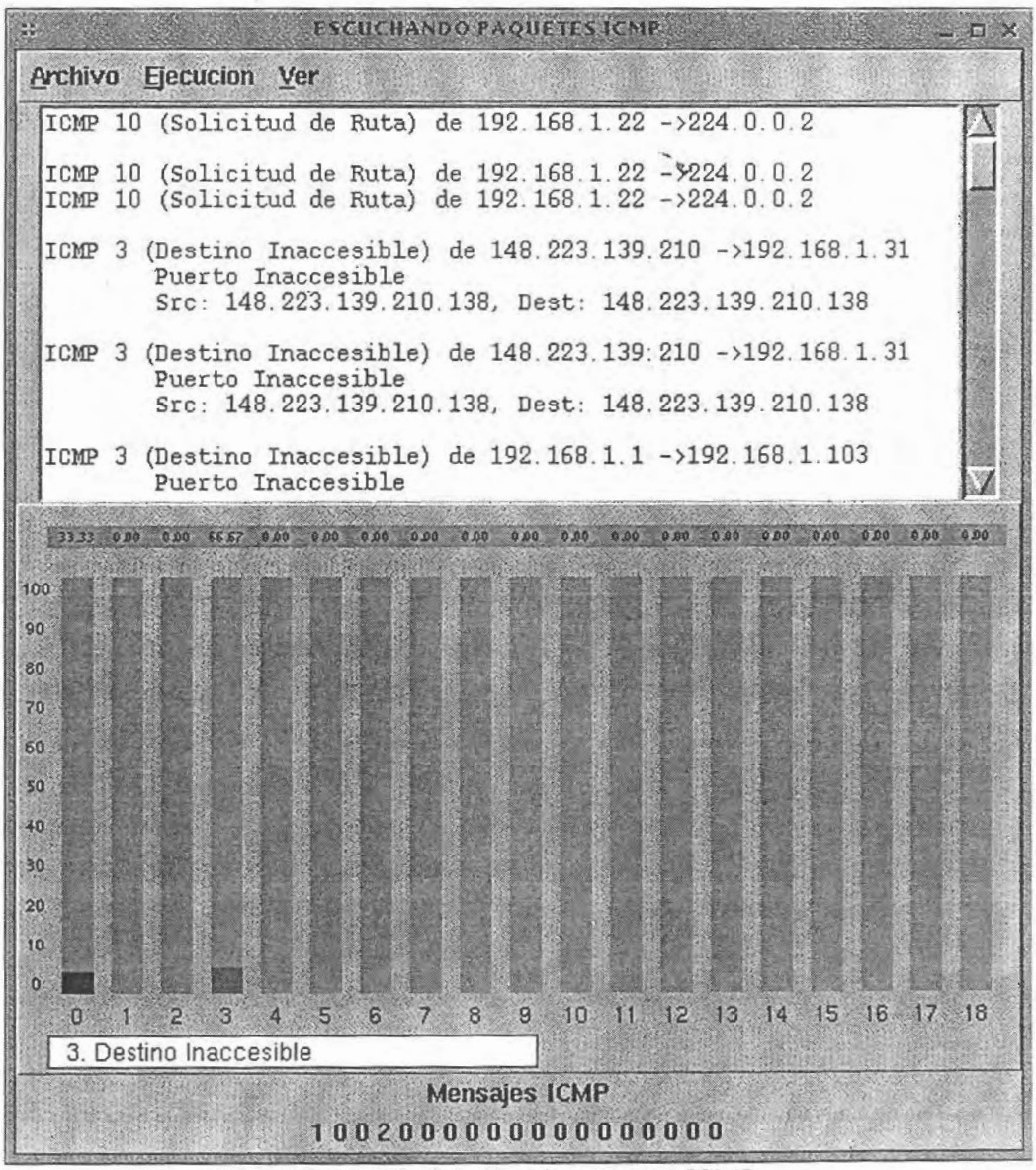


Figura 3.3. Escuchando paquetes ICMP

La gráfica presenta sólo 18 tipos de mensajes ICMP de los 40 que se presentaron en el capítulo 1, debido a que a partir del tipo 19 corresponden a mensajes que normalmente

no van a circular en una LAN Ethernet sobre IPv4, que es para el tipo de red que está diseñada este sistema de monitoreo.

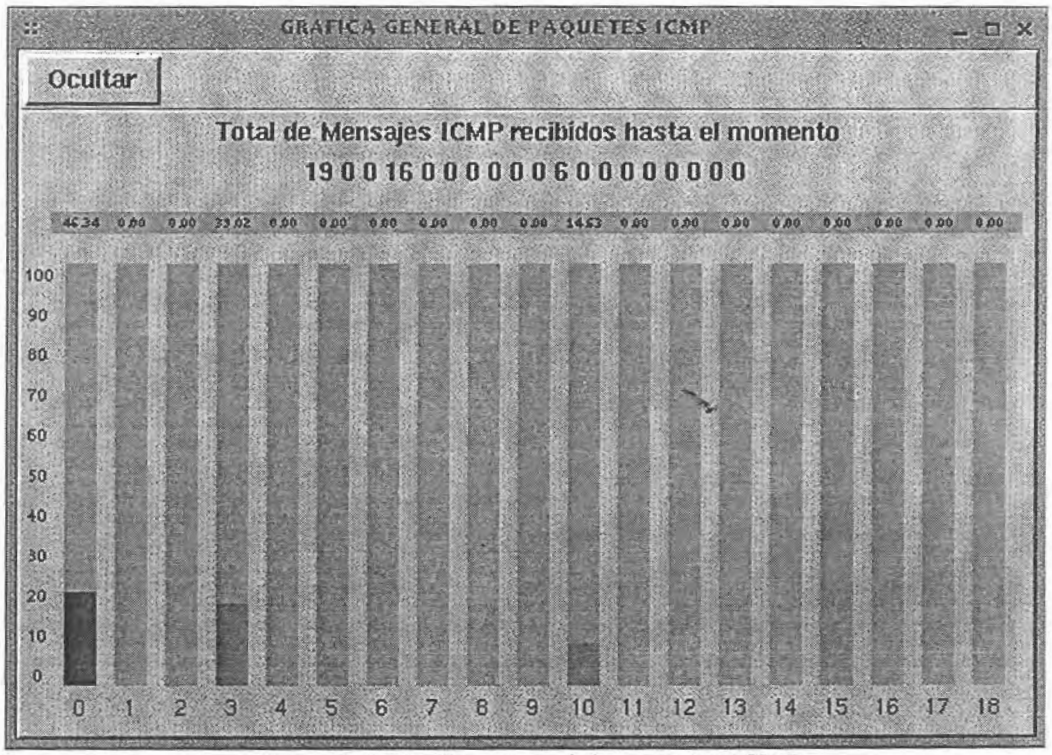


Figura 3.4. Grafica general de paquetes ICMP

La pantalla que se presentan al capturar paquetes ARP se muestra en la figura 3.5, ésta sólo muestra dos barras debido a que existen solamente dos tipos de mensajes, Solicitud y Respuesta ARP. Esta aplicación permite visualizar también una gráfica general de los paquetes escuchados, dicha gráfica se presenta en la figura 3.6.

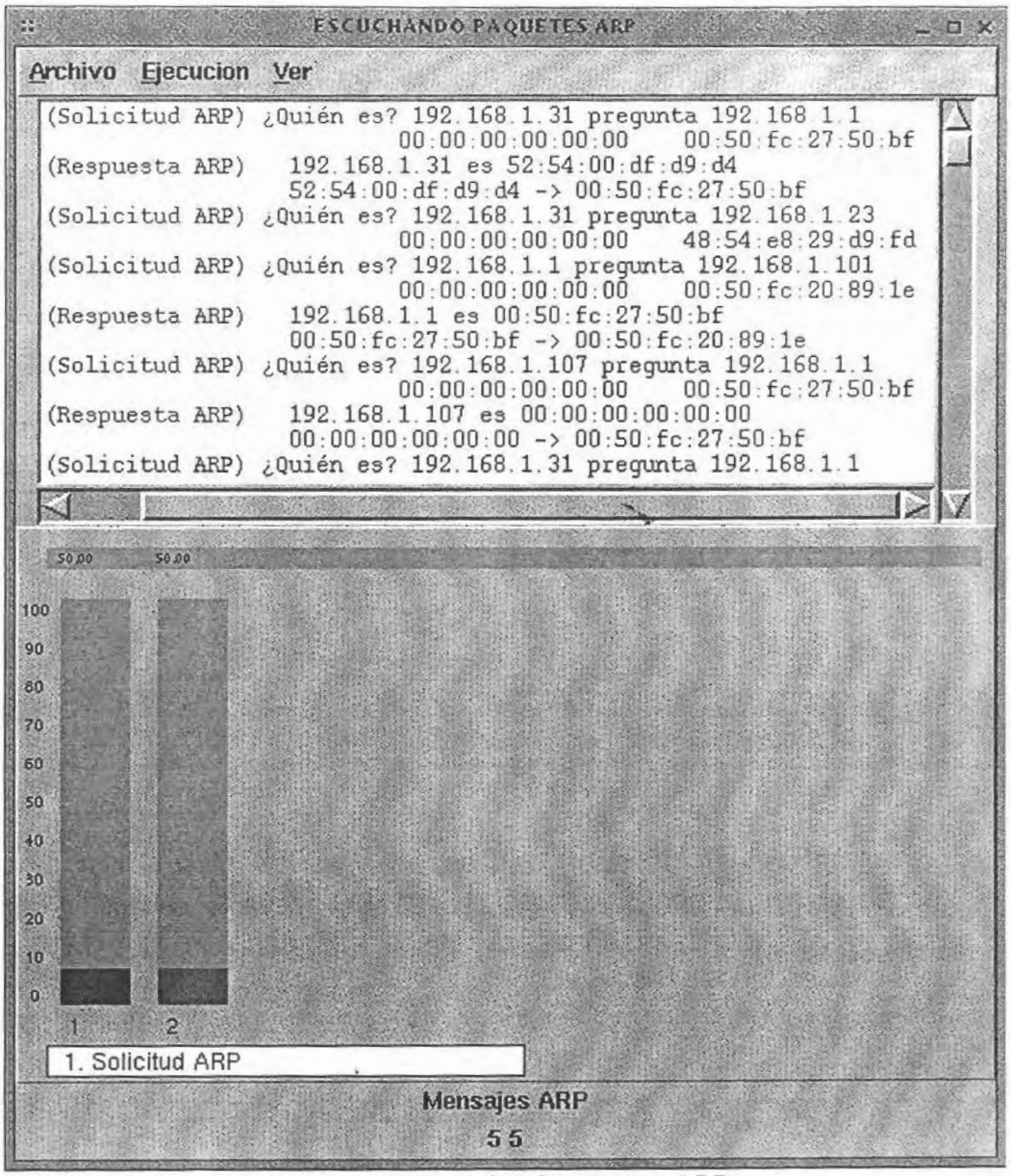


Figura 3.5. Escuchando paquetes ARP

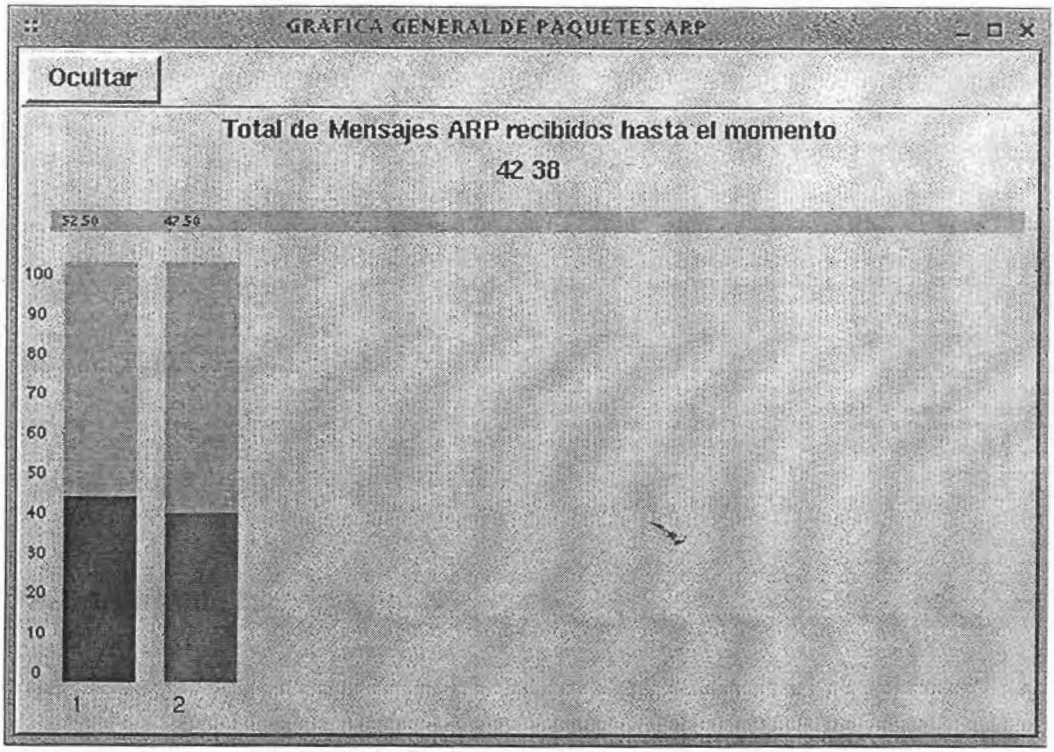


Figura 3.6. Grafica general de paquetes ARP

Para RARP las pantallas presentadas son parecidas a las mostradas en la figura 3.5 y figura 3.6, con la única diferencia de que estas pantallas presentan los tipos de mensaje Solicitud y Respuesta RARP. No se tienen ejemplos de gráficas realizadas, debido a que la red en la que fue probado el sistema de monitoreo no existe ninguna computadora que requiera el uso de este protocolo.

3.4 ESPECIFICACIONES DE HARDWARE Y SOFTWARE

El hardware utilizado en la implementación de este proyecto ha sido una máquina con procesador AMD K6III a 400Mhz con 64 Mb RAM y tarjeta de red DAVICOM 9102 PCI 10/100; el sistema operativo bajo el cual se ha desarrollado es Linux con kernel 2.2.15-4mdk y los lenguajes de programación que han sido utilizados son: C para los módulos que realizan la tarea de captura de paquetes y Tcl/Tk para la interfaz gráfica.

Capítulo 4

AVICAR PARA REDES IPv6

En el presente capítulo se describen los principales cambios que presenta el protocolo IPv6 sobre IPv4, así como los protocolos ICMPv6 y ND, una vez presentada dicha información se describen las estructuras y funciones requeridas para el funcionamiento de AVICAR en redes IPv6.

4.1 PROTOCOLO DE INTERNET VERSIÓN 6 (IPv6)

IPv6 o IPng[1] es el protocolo diseñado por la IETF[2] con el fin de reemplazar al protocolo IP versión 4.

Los cambios introducidos al IPv6 pueden agruparse en cinco categorías [15]:

- Capacidades de direccionamiento más amplias.
  IPv6 aumenta el tamaño de las direcciones IP de 32 bits a 128 bits, para soportar niveles adicionales de jerarquía de direcciones, un número mucho más grande de nodos direccionables y una autoconfiguración más simple de direcciones. Así mismo, se define un nuevo tipo de direcciones llamado "dirección anycast", usado para enviar un paquete a un miembro de un grupo de nodos.

[1] *Next Generation Internet Protocol.*- Protocolo de Internet de la Próxima Generación
[2] *Internet Engineering Task Force.*- Fuerza de Tarea de Ingeniería de Internet

- Formato de encabezado más simple.
  Algunos campos del encabezado IPv4 han sido removidos o hecho opcionales, para reducir el costo de procesamiento en el manejo de paquetes y para limitar el costo del ancho de banda del encabezado de IPv6.

- Soporte mejorado para Extensiones y Opciones.
  Los cambios en la forma en que las opciones del encabezado IP son codificadas permiten un reenvío más eficiente, menos estrictos en la longitud límite de las opciones, y una mayor flexibilidad para introducir nuevas opciones en el futuro.

- Capacidad para Asignación de Flujos.
  Una nueva capacidad es agregada para permitir la asignación de paquetes que corresponden a un tráfico particular "flujos" para el cual el transmisor solicita un manejo especial, tal como calidad del servicio diferente o servicio en "tiempo real".

- Capacidad de Autenticación y Privacidad.
  Extensiones para soportar autenticación, integridad, y (opcional) confidencialidad de datos son especificados por IPv6.

4.1.1 ESPECIFICACIONES BÁSICAS DE IPV6

Para explicar las diferencias entre un datagrama IPv6 y uno IPv4, se muestra en la figura 4.1 la descripción del encabezado de un datagrama IPv4, mostrando en color gris los campos que han sido modificados y en blanco los que han desaparecido en IPv6 [33].

<table>
<tr><td colspan="32">0 4 8 16 19 24 31</td></tr>
<tr><td colspan="4">VERS</td><td colspan="4">LONGE</td><td colspan="8">TIPO DE SERVICIO</td><td colspan="16">LONGITUD TOTAL</td></tr>
<tr><td colspan="16">IDENTIFICACIÓN</td><td colspan="3">BAN-
DERAS</td><td colspan="13">DESPLAZAMIENTO DE
FRAGMENTO</td></tr>
<tr><td colspan="8">TIEMPO DE VIDA</td><td colspan="8">PROTOCOLO</td><td colspan="16">SUMA DE VERIFICACIÓN
DEL ENCABEZADO</td></tr>
<tr><td colspan="32">DIRECCIÓN IP DE LA FUENTE DE 32 BITS</td></tr>
<tr><td colspan="32">DIRECCIÓN IP DEL DESTINO DE 32 BITS</td></tr>
<tr><td colspan="24">OPCIONES IP (SI LAS HAY)</td><td colspan="8">RELLENO</td></tr>
</table>

Figura 4.1. Formato del encabezado de un datagrama IPv4

El motivo fundamental por el que los campos son eliminados, es la innecesaria redundancia. En IPv4 se da la misma información de varias formas. Uno de estos casos es la SUMA DE VERIFICACIÓN DEL ENCABEZADO, pues existen otros mecanismos de encapsulado que realizan esta función (IEEE 802MAC, framing PPP, capa de adaptación ATM, etc.). En el caso del campo DESPLAZAMIENTO DE FRAGMENTO, éste es

eliminado, debido a que el mecanismo por el que se realiza la fragmentación de los paquetes en IPv6 es totalmente modificado. En IPv6 los ruteadores no realizan la tarea de fragmentación de los paquetes, sino que se realiza en el nodo inicial.

Algunos campos son renombrados:

- LONGITUD TOTAL → LONGITUD DE CARGA ÚTIL, que es la longitud de los datos y puede ser de hasta 65,536 bytes.
- TIEMPO DE VIDA → LÍMITE DE SALTOS.
- PROTOCOLO → PRÓXIMO ENCABEZADO, esto es porque en lugar de usar encabezados de longitud variable se emplean encabezados encadenados, éstos no son examinados en cada nodo sino sólo en el nodo o nodos finales.

Los nuevos campos son:

- CLASE DE TRÁFICO, llamado también PRIORIDAD, o simplemente CLASE. Es más o menos igual a TOS en IPv4.
- ETIQUETA DE FLUJO, permite tráfico con requisitos de tiempo real y asignación de flujos.

Estos dos campos son los que permiten las características fundamentales e intrínsecas de IPv6: Calidad de Servicio (QoS), Clase de Servicio (CoS), y un mecanismo de Control de flujo para la asignación de prioridades diferenciadas según los tipos de servicios.

El encabezado de un datagrama IPv6 [15], tiene el siguiente formato (figura 4.2):

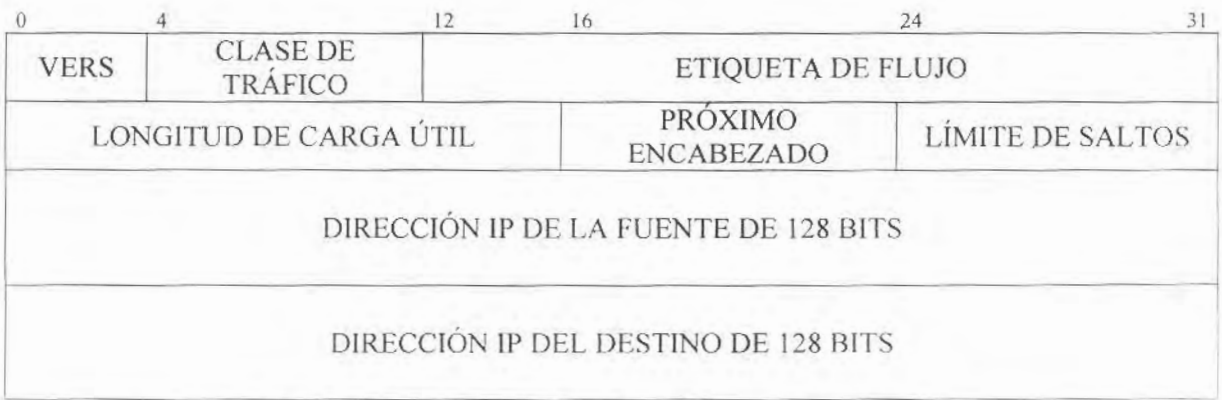


Figura 4.2. Formato del encabezado de un datagrama IPv6

La longitud de este encabezado es de 40 bytes, el doble que en el caso de IPv4, pero con muchas ventajas, al haberse eliminado campos redundantes.

En IPv6, la información de la capa de Internet es codificada en encabezados separados que pueden ser colocados entre el encabezado IPv6 y el encabezado de la capa superior. Hay un número pequeño de tales encabezados de extensión, cada uno identificado por un valor distinto en PRÓXIMO ENCABEZADO. Un encabezado IPv6 puede tener cero, uno o más encabezados de extensión, cada uno identificado por el campo PRÓXIMO ENCABEZADO del encabezado precedente. Los encabezados de extensión no son analizados en cada nodo de la ruta, sino sólo en el nodo o nodos finales.

La MTU, debe de ser como mínimo, de 1280 bytes, aunque se recomiendan tamaños superiores a 1500 bytes.

4.1.2 DIRECCIONAMIENTO DE IPv6

Las direcciones identifican interfaces individuales o conjuntos de interfaces, éstas se asignan a interfaces no a nodos. A una interfaz se le podrían asignar varias direcciones IPv6 de cualquier tipo, éstas se clasifican en [14]:

- *Unicast*: la dirección de destino especifica una sola interfaz (computadora, servidor o ruteador). Un paquete enviado a una dirección *unicast* es entregado sólo a la interfaz identificada con dicha dirección. Es el equivalente a las direcciones IPv4.
- *Anycast*: la dirección de destino identifica a un conjunto de interfaces; un paquete enviado a una dirección *anycast* es ruteado hacía el grupo de interfaces y entregado a un solo miembro de dicho grupo, el más cercano de acuerdo a la medida de distancia de los protocolos de ruteo.
- *Multicast*: la dirección identifica a un conjunto de interfaces, posiblemente en múltiples localidades. Una copia del datagrama deberá entregarse a cada miembro del grupo identificado con esa dirección.

En IPv6 no hay direcciones *broadcast*, su función ha sido reemplazada con las direcciones *multicast*. Una dirección *unicast* se refiere a una simple interfaz. Puesto que cada interfaz pertenece a un simple nodo, cualquiera de las direcciones *unicast* de las interfaces de los nodos puede ser usado como un identificador para el nodo.

Todas las interfaces deben tener por lo menos una dirección *unicast* para enlace local. A cada interfaz se le pueden asignar múltiples direcciones IPv6 de cualquier tipo (*unicast, anycast, multicast*).

4.1.3 REPRESENTACIÓN DE LAS DIRECCIONES IPV6

Hay tres formas convencionales para representar las direcciones IPv6:

- La forma predilecta es x:x:x:x:x:x:x:x, donde las x's son los valores hexadecimales de los ocho campos de 16 bits que forman las direcciones. Ejemplos:

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

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

  No es necesario escribir los ceros en un campo individual, pero debe haber por lo menos un número en cada campo, excepto para el caso descrito en el siguiente punto.

- Debido a algunos métodos de ciertos estilos de asignación de direcciones IPv6, será común para las direcciones contener cadenas largas de ceros. Por lo que existe la posibilidad de usar sintácticamente :: para representarlos. El uso de :: indica múltiples grupos de 16 bits de ceros. Dicho símbolo podrá aparecer una sola vez en cada dirección. Los :: pueden también ser usados para comprimir varios campos de 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	una dirección de lazo
0:0:0:0:0:0:0:0	una dirección no especificada

  pueden ser representadas como:

1080::8:800:200C:417A	una dirección *unicast*
FF01::101	una dirección *multicast*
::1	una dirección de lazo
::	una dirección no especificada

- Para redes con computadoras IPv4 e IPv6 se puede utilizar la siguiente sintaxis: x:x:x:x:x:x:d.d.d.d, donde las x's representan valores hexadecimales de las seis partes más significativas (de 16 bits cada una) que componen la dirección y las d's. son valores decimales de las 4 partes menos significativas (de 8 bits cada una), de la representación estándar del formato de direcciones IPv4.

  Ejemplos:

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

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

o en su forma comprimida:

::13.1.68.3
:: FFFF:129.144.52.38

4.1.4 REPRESENTACIÓN DE LOS PREFIJOS DE DIRECCIONES

La representación de los prefijos de direcciones IPv6 es similar a la forma en que los prefijos de direcciones IPv4 son escritos en notación CIDR[3]. Un prefijo de dirección IPv6 es representado por la notación:

dirección-ipv6 / longitud-prefijo

donde:

dirección-ipv6	es una dirección IPv6 en cualquiera de las notaciones explicadas anteriormente.
longitud-prefijo	es un valor decimal que especifica cuántos de los bits contiguos más a la izquierda de la dirección abarcan el prefijo.

Por ejemplo, las siguientes son representaciones válidas 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 válidas del prefijo de arriba:

12AB:0:0:CD3/60	Se pueden quitar ceros, pero no dentro de cualquier fragmento de 16 bits de la dirección
12AB::CD30:/60	La dirección a la izquierda de "/" se expande a 12AB:0000:0000:0000:0000:0000:0000:CD30
12AB::CD3/60	La dirección a la izquierda de "/" se expande a 12AB:0000:0000:0000:0000:0000:0000:CD3

[3] *Classless Inter-Domain Routing* .- Ruteo sin clases entre dominios, [RFC1481] [RFC1517, 1518, 1519], con él los ruteadores reducen el tamaño de sus tablas colapsando juntas varias subredes con el mismo prefijo.

Cuando se escriben ambas, una dirección de un nodo y un prefijo de la dirección del nodo (por ejemplo: el prefijo de la subred del nodo), los dos pueden ser combinados de la siguiente forma:

La dirección del nodo	12AB:0:0:CD30:123:4567:89AB:CDEF
Y el número de subred	12AB:0:0:CD30::/60
Puede ser abreviado como	12AB:0:0:CD30:123:4567:89AB:CDEF/60

4.2 *PROTOCOLO DE CONTROL DE MENSAJES DE INTERNET VERSIÓN 6*

El ICMP para IPv4 ha sido actualizado para permitir su uso bajo IPv6, dando como resultado el ICMPv6, al cual se le ha asignado el valor de 58, para el campo de PRÓXIMO ENCABEZADO. ICMPv6 es parte integral de IPv6 y debe ser incorporado por completo a cualquier implementación de IPv6.

ICMPv6 es utilizado por IPv6 para reportar errores generados durante el procesamiento de los paquetes, así como para diagnósticos. La figura 4.3 muestra el formato general de los mensajes ICMPv6 [17].

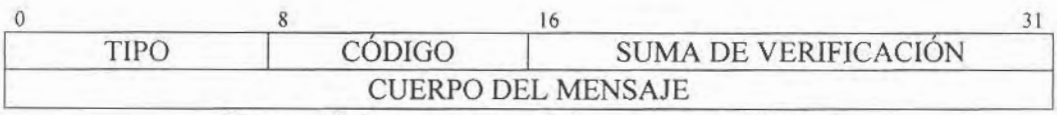


Figura 4.3. Formato general de un mensaje ICMPv6

En donde:

- TIPO, indica el tipo de mensaje, y su valor determina el formato del resto de la cabecera,
- CÓDIGO, depende del tipo de mensaje, y es utilizado para crear un nivel adicional de jerarquía para la clasificación del mensaje, y
- SUMA DE VERIFICACIÓN, permite detectar errores en el mensaje ICMPv6 y partes del encabezado IPv6.

Los mensajes ICMPv6 se agrupan en dos tipos o clases: mensajes de error y mensajes informativos. Los mensajes de error tienen cero en el bit más significativo del campo TIPO, por lo que sus valores se sitúan entre 0 y 127. Los valores de los mensajes informativos varían entre 128 y 255. Los mensajes definidos por la especificación básica se muestran en la tabla 4.1.

Tabla 4.1. Tipos de mensaje ICMPv6

Tipo	Código	Descripción de mensaje ICMPv6	Referencias
1		Destino inaccesible	RFC 2463
	0	Sin ruta hacía el destino	
	1	Comunicación prohibida administrativamente	
	2	No es un vecino	
	3	Dirección inalcanzable	
	4	Puerto inalcanzable	
2		Paquete demasiado grande	RFC 2463
3		Tiempo excedido	RFC 2463
	0	Límite de saltos excedido	
	1	Tiempo de reensamblado excedido	
4		Problema de parámetros	RFC 2463
	0	Campo erróneo en el encabezado	
	1	Tipo de Próximo encabezado desconocido	
	2	Opción IPv6 desconocida	
128		Solicitud de eco	RFC 2463
129		Respuesta de eco	RFC 2463
130		Búsqueda de nodos Multicast	RFC 2710
131		Reporte de nodo Multicast	RFC 2710
132		Abandono de nodo Multicast	RFC 2710
133		Solicitud de ruteador	RFC 2461
134		Anunciación de ruteador	RFC 2461
135		Solicitud de vecino	RFC 2461
136		Anunciación de vecino	RFC 2461
137		Redirección	RFC 2461
138		Renumeración de ruteador	RFC 2894
139		Solicitud de Información de nodo ICMPv6	En desarrollo
140		Respuesta de Información de nodo ICMPv6	En desarrollo
141		Solicitud del mensaje de descubrimiento inverso de vecino	RFC 3122
142		Aviso del mensaje de descubrimiento inverso de vecino	RFC 3122

4.2.1 CONSIDERACIONES DE SEGURIDAD

El intercambio de paquetes del protocolo ICMPv6 puede ser autentificado usando el encabezado de autenticación IPv6. Para esto, si existe una asociación de uso del encabezado de autenticación IPv6 para la dirección a la que se va a enviar el mensaje, el nodo transmisor debe incluir un encabezado de autenticación en el mensaje ICMPv6 a

enviar. Las asociaciones de seguridad deben haber sido creadas a través de la configuración manual o a través de la operación de algún protocolo administrador de llaves.

Los encabezados de autenticación recibidos en paquetes ICMPv6 se deben verificar con exactitud y los paquetes con autenticación incorrecta deben ser ignorados y descartados.

El administrador del sistema puede configurar un nodo para que ignore cualquier mensaje ICMPv6 que no esté autenticado.

4.2.2 POSIBLES ATAQUES USANDO ICMP

Los mensajes ICMPv6 pueden estar sujetos a varios ataques, estos ataques y su prevención son descritos brevemente a continuación:

- Los mensajes ICMPv6 pueden estar sujetos a acciones que intentan provocar que el receptor crea que el mensaje vino de una fuente diferente que el origen del mensaje. La protección en contra de este ataque puede conseguirse aplicando el mecanismo de Autenticación IPv6 al mensaje ICMPv6.
- Los mensajes ICMPv6 pueden estar sujetos a acciones que intentan provocar que el mensaje o respuesta vaya a un destino diferente que la intención del mensaje original. El cálculo de la Suma de Verificación ICMPv6 provee un mecanismo de protección en contra de cambios por un interceptor malicioso en la dirección destino y fuente del paquete IP que transporta el mensaje, el campo Suma de Verificación proporcionado es protegido contra cambios por autenticación o encriptación del mensaje ICMPv6.
- Los mensajes ICMPv6 pueden estar sujetos a cambios en los campos del mensaje, o en su carga útil. La autenticación o encriptación del mensaje ICMPv6 es una protección en contra de tales acciones.
- Los mensajes ICMPv6 pueden ser usados como intento para ejecutar ataques de negación de servicio enviando paquetes IP erróneos consecutivos, esto podría protegerse mediante un mecanismo de limitación de velocidad de error ICMPv6.

Actualmente se está trabajando en nuevos tipos de mensajes, siendo uno de ellos el definido en un borrador de IETF (draft-ieft-ipngwg-icmp-name-lookups-05.txt), que permitirá solicitar a una máquina información completa como su nombre de dominio completamente cualificado (Tipo 139).

4.3 *PROTOCOLO DE DESCUBRIMIENTO DE VECINOS*

En IPv6, el protocolo semejante a ARP en IPv4, es el protocolo ND. Este protocolo consiste en el mecanismo por el cual un nodo que se incorpora a una red, descubre la presencia de otros en su mismo enlace, determina sus direcciones en la capa de enlace, localiza los ruteadores y mantiene la información de conectividad acerca de las rutas a los vecinos activos [17].

El protocolo ND se emplea también para mantener limpios los caches donde se almacena la información relativa al contexto de la red a la que está conectada un anfitrión (servidor o ruteador), y para detectar cualquier cambio en la misma. Si un ruteador o una ruta falla, el servidor buscará alternativas funcionales.

ND emplea los mensajes ICMPv6 para algunos de sus servicios, este protocolo es bastante completo y sofisticado, ya que es la base para permitir el mecanismo de autoconfiguración en IPv6. Define varios mecanismos, entre ellos: descubrir ruteadores, prefijos y parámetros, autoconfiguración de direcciones, resolución de direcciones, determinación del siguiente salto, detección de nodos inalcanzables, detección de direcciones duplicadas o cambios, redirección, balanceo de carga entrante, direcciones *anycast,* y anunciación de proxies.

ND define cinco tipos de paquetes ICMPv6:

- Solicitud de ruteador (Tipo 133). Generado por una interfaz cuando es activada, para pedir a los ruteadores que se anuncien inmediatamente.
- Anunciación de ruteador (Tipo 134). Generado por los ruteadores periódicamente (entre cada 4 y 1800 segundos) o como consecuencia del tipo anterior, a través de multicast, para informar de su presencia así como de otros parámetros de enlace y de Internet (prefijos, tiempo de vida, configuración de direcciones, tamaño máximo de la unidad de transmisión o MTU, etc.). Es importante para permitir la renumeración. Este tipo de mensaje contiene prefijos que son usados para determinar la red a la que está conectado un nodo y/o configuración de direcciones, para dar a conocer el valor de límite de saltos.
- Solicitud de vecino (Tipo 135). Generado por los nodos para determinar la dirección en la capa de enlace de sus vecinos, para verificar que el nodo vecino es alcanzable, así como para detectar las direcciones duplicadas.
- Anunciación de vecino (Tipo 136). Generado por los nodos como respuesta a la Solicitud de vecino, o bien para indicar cambios de direcciones en la capa de enlace.

- Redirección (Tipo 137). Generado por los ruteadores para informar a las máquinas de un mejor salto para llegar a un destino determinado. Es en parte semejante a ICMP Redireccionar (Tipo 5).

El protocolo ND tiene varias ventajas frente a los mecanismos existentes en IPv4:

- El descubrimiento de ruteadores es parte de la base del protocolo, no se tiene que recurrir a los protocolos de ruteo.
- La anunciación del ruteador incluye las direcciones de la capa de enlace, debido a esto no es necesario ningún intercambio adicional de paquetes para su resolución.
- La anunciación del ruteador incluye los prefijos para el enlace, por lo que no hay necesidad de un mecanismo adicional para configurar la máscara de red.
- La anunciación de un ruteador permite la autoconfiguración de direcciones.
- Los ruteadores pueden anunciar a los servidores del mismo enlace la MTU.
- Las redirecciones contienen la dirección de la capa de enlace del nuevo salto, lo que evita la necesidad de una resolución de dirección adicional.
- Se pueden asignar múltiples prefijos al mismo enlace y por defecto los servidores aprenden todos los prefijos por la anunciación del ruteador. Sin embargo, los ruteadores pueden ser configurados para omitir parte o todos los prefijos en la anunciación, de forma que las máquinas consideren que los destinos están fuera del enlace y por lo tanto envíen el tráfico a los ruteadores, quien a su vez los redireccionará según corresponda.
- A diferencia de IPv4, en IPv6 el receptor de una redirección asume que el siguiente salto está en el mismo enlace. Se prevé una gran utilidad en el sentido de no ser deseable o posible que los nodos conozcan todos los prefijos de los destinos en el mismo enlace.
- La detección de vecinos inalcanzables es parte de la base de mejoras para la robustez en la entrega de paquetes frente a fallos en ruteadores, fallas parciales o particiones de enlaces, nodos que cambian sus direcciones, nodos móviles, etc.
- A diferencia de ARP, ND puede detectar fallos de la mitad del enlace, es decir, con conectividad en un solo sentido, evitando el tráfico hacia ellos.
- El uso de direcciones de enlace local para identificar ruteadores, permite a las máquinas que mantengan su asociación con los mismos, en el caso de que se realice una renumeración para usar nuevos prefijos globales.
- El límite de saltos es siempre igual a 255, lo que evita que haya envíos accidentales o intencionados desde máquinas fuera del enlace, dado que los ruteadores decrementan automáticamente este campo en cada salto. En IPv4 los transmisores pueden enviar tanto mensajes ICMP Redireccionar (Tipo 5), como mensajes de Anunciación de Ruteador (Tipo 134).

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

Hasta aquí se han presentado los protocolos IPv6, a continuación se hablará de las funciones y estructuras definidas en C que dan soporte a estos protocolos, tomándolas como base será posible hacer los ajustes necesarios a AVICAR para obtener su funcionalidad sobre redes IPv6.

4.4 FUNCIONES Y ESTRUCTURAS DE C PARA IPV6

4.4.1 FAMILIA DE PROTOCOLOS Y FAMILIA DE DIRECCIONES IPv6

En la implementación de AVICAR se utilizó la familia de direcciones AF_INET, en IPv6 surge una nueva familia de direcciones llamada AF_INET6, definida en <sys/socket.h>. La definición AF_INET6 distingue entre la estructura de datos sockaddr_in original, y la estructura de datos nueva sockaddr_in6 [18]. Aunque también se puede utilizar la familia de protocolo, PF_INET6, definida en <sys/socket.h>. Por la equivalencia:

```
#define PF_INET6        AF_INET6
```

La PF_INET6 es usada como primer argumento en la función socket(), para indicar que un socket IPv6 está siendo creado.

4.4.2 FUNCIONES DE SOCKET

Las aplicaciones llaman a la función socket() para crear un descriptor de socket que representa el punto de comunicación. Los argumentos de la función socket() le dicen al sistema cuál protocolo se va a utilizar, y qué formato de estructura de direcciones será usado en funciones subsecuentes. Por ejemplo, para crear un socket IPv4/TCP, las aplicaciones hacen una llamada:

```
s = socket(PF_INET, SOCK_STREAM, 0);
```

Las aplicaciones pueden crear sockets IPv6/TCP usando simplemente la constante PF_INET6 en lugar de PF_INET en el primer argumento. Por ejemplo, para crear un socket IPv6/TCP, las aplicacines hacen una llamada:

```
s = socket(PF_INET6, SOCK_STREAM, 0);
```

Una vez que la aplicación ha creado un socket PF_INET6, debe usar la estructura de dirección sockaddr_in6 cuando pasen direcciones al interior del sistema. El sistema usara esta estructura de dirección para regresar direcciones a las aplicaciones que están usando sockets PF_INET6. Una de las funciones que regresan una dirección del sistema a una aplicación es:

recvfrom()

La sintaxis de esta función no cambia para soportar IPv6.

4.4.3 FUNCIÓN PARA CONVERSIÓN DE DIRECCIONES

En el módulo que captura paquetes ICMP de AVICAR se ha hecho uso de la función inet_ntoa() para convertir una dirección IPv4 a tipo texto. Las aplicaciones IPv6 necesitan una función similar, que es inet_ntop(), la cual convierte ambos tipos de direcciones IPv4 e IPv6.

```
#include <sys/socket.h>
#include <arpa/inet.h>

const char *inet_ntop(int fam, const void *org, char *dest, size_t tam);
```

En donde:

- fam, especifica la familia de direcciones, que puede ser AF_INET o AF_INET6.
- org, es un puntero al buffer que va a almacenar una dirección IPv4 si el argumento es AF_INET, o una dirección IPv6 si el argumento es AF_INET6,
- des, es un puntero a un buffer donde la función almacenará el resultado de tipo texto, la aplicación debe especificar un buffer ya que no debe ser NULL. Para direcciones IPv6, el buffer debe ser de por lo menos 46 bytes y para direcciones IPv4 debe ser de por lo menos 16 bytes. Con el fin de permitir a las aplicaciones declarar fácilmente los buffers del tamaño adecuado, son definidas dos constantes en <netinet/in.h>

  ```
  #define INET_ADDRSTRLEN   16
  #define INET6_ADDRSTRLEN  46
  ```
- tam, especifica el tamaño del buffer.

4.4.4 ESTRUCTURAS DEFINIDAS PARA ENCABEZADOS VERSIÓN 6

Debido a que los módulos de AVICAR encargados de capturar los paquetes que circulan por la red examinan los campos de los encabezados de los datagramas IPv4, es

necesario especificar las estructuras de datos que permiten almacenar los encabezados de protocolos versión 6 [13].

4.4.4.1 Estructura del encabezado IPv6

- **struct ip6_hdr**, declarada en <netinet/ip6.h> y define el formato del encabezado IPv6.

```
struct ip6_hdr {
 union {
  struct ip6_hdrctl {
   uint32_t ip6_un1_flow;  /* 24 bits de etiqueta de flujo */
   uint16_t ip6_un1_plen;  /* longitud de carga útil */
   uint8_t  ip6_un1_nxt;   /* próximo encabezado */
   uint8_t  ip6_un1_hlim;  /* límite de saltos */
  } ip6_un1;
  uint8_t ip6_un2_vfc;     /* 4 bits version, 4 bits prioridad */
 } ip6_ctlun;
 struct in6_addr ip6_src;    /* dirección origen */
 struct in6_addr ip6_dst;    /* dirección destino */
};
```

Debido a que IPv6 maneja lo que se llama Encabezados de extensión definidos por el campo Próximo encabezado, es necesario implementar la parte que analice estos encabezados, para lo cual se presentan los valores que puede tomar el campo Próximo encabezado y los formatos de los encabezados de extensión.

- **Valores de próximo encabezado IPv6**, son definidos en <netinet/in.h>

```
#define IPPROTO_HOPOPTS      0     /* opciones de examinado y proceso
                                      salto a salto IPv6*/
#define IPPROTO_IPV6         41    /* encabezado IPv6 */
#define IPPROTO_ROUTING      43    /* encabezado Ruteo IPv6 */
#define IPPROTO_FRAGMENT     44    /* encabezado de Fragmentación IPv6*/
#define IPPROTO_ESP          50    /* encapsulado de carga útil segura*/
#define IPPROTO_AH           51    /* encabezado de autenticación */
#define IPPROTO_ICMPV6       58    /* ICMPv6 */
#define IPPROTO_NONE         59    /* no próximo encabezado IPv6 */
#define IPPROTO_DSTOPTS      60    /* opciones de Destino IPv6 */
```

Las implementaciones también definen IPPROTO_IP con valor de 0, esto no debe ser un problema puesto que IPPROTO_IP es usado sólo con sockets IPv4 e IPPROTO_HOPOPTS sólo con sockets IPv6.

- **Encabezados de extensión IPv6**, están definidos en <netinet/ip6.h>, excepto el encabezado de encapsulado de carga útil segura y el encabezado de autenticación.

```
/* Opciones de examinado y proceso salto a salto IPv6 */
struct ip6_hbh {
 uint8_t  ip6h_nxt;      /* próximo encabezado */
 uint8_t  ip6h_len;      /* longitud en unidades de 8 bytes */
  /* seguido por opciones */
};

/* Opciones de Destino IPv6 */
struct ip6_dest {
 uint8_t  ip6d_nxt;      /* próximo encabezado */
 uint8_t  ip6d_len;      /* longitud en unidades de 8 bytes */
  /* seguido por opciones */
};

/* Encabezado de ruteo */
struct ip6_rthdr {
 uint8_t  ip6r_nxt;      /* próximo encabezado */
 uint8_t  ip6r_len;      /* longitud en unidades de 8 bytes */
 uint8_t  ip6r_type;     /* tipo de ruteo */
 uint8_t  ip6r_segleft;  /* segmento salido */
  /* seguido por un dato específico de tipo de ruteo */
};

/* Tipo 0 encabezado de Ruteo */
struct ip6_rthdr0 {
 uint8_t  ip6r0_nxt;     /* próximo encabezado*/
 uint8_t  ip6r0_len;     /* longitud en unidades de 8 bytes */
 uint8_t  ip6r0_type;    /* siempre cero */
 uint8_t  ip6r0_segleft; /* segmento salido */
 uint8_t  ip6r0_reserved; /* campo reservado */
 uint8_t  ip6r0_slmap[3]; /* mapa de bit riguroso/no riguroso */
 struct in6_addr  ip6r0_addr[1]; /* superior a 23 direcciones */
};

/* encabezado de fragmento*/
struct ip6_frag {
 uint8_t  ip6f_nxt;      /* próximo encabezado*/
 uint8_t  ip6f_reserved; /* campo reservedo */
 uint16_t ip6f_offlg;    /* desplazado, reservado y bandera*/
 uint32_t ip6f_ident;    /* identificación */
};
```

4.4.4.2 Estructura del encabezado ICMPv6

- **struct icmp6_hdr**, declarada en <netinet/icmp6.h> y define el formato del encabezado ICMPv6.

```
struct icmp6_hdr {
 uint8_t    icmp6_type;  /* campo tipo */
 uint8_t    icmp6_code;  /* campo código */
 uint16_t   icmp6_cksum; /* campo suma de verificación */
 union {
  uint32_t  icmp6_un_data32[1]; /* campo tipo específico */
  uint16_t  icmp6_un_data16[2]; /* campo tipo específico */
  uint8_t   icmp6_un_data8[4]; /* campo tipo específico */
 } icmp6_dataun;
};
```

- **Valores de Tipo y Código ICMPv6**, estas constantes están definidas también en <netinet/icmp6.h>.

```
#define ICMP6_DST_UNREACH            1
#define ICMP6_PACKET_TOO_BIG         2
#define ICMP6_TIME_EXCEEDED          3
#define ICMP6_PARAM_PROB             4

#define ICMP6_INFOMSG_MASK  0x80  /* todos los mensajes informativos */

#define ICMP6_ECHO_REQUEST           128
#define ICMP6_ECHO_REPLY             129

#define ICMP6_MEMBERSHIP_QUERY   130
#define ICMP6_MEMBERSHIP_REPORT 131
#define ICMP6_MEMBERSHIP_REDUCTION  132

#define ICMP6_DST_UNREACH_NOROUTE       0     /* sin ruta al destino */
#define ICMP6_DST_UNREACH_ADMIN         1     /* comunicación prohibida */
                                              /* administrativamente */
#define ICMP6_DST_UNREACH_NOTNEIGHBOR 2       /* no es vecino */
#define ICMP6_DST_UNREACH_ADDR          3     /* dirección inalcanzable */
#define ICMP6_DST_UNREACH_NOPORT        4     /* puerto inalcanzable */

#define ICMP6_TIME_EXCEED_TRANSIT       0     /* límite de salto == 0 in
                                                 tránsito */
#define ICMP6_TIME_EXCEED_REASSEMBLY    1     /* tiempo de reensamblado
                                                 excedido*/

#define ICMP6_PARAMPROB_HEADER          0     /* campo erróneo en el
                                                 encabezado */
#define ICMP6_PARAMPROB_NEXTHEADER      1     /* próximo encabezado
                                                 desconocido */
```

```
#define ICMP6_PARAMPROB_OPTION          2       /* opcion IPv6 desconocida */
```

Debido a que el protocolo ND hace uso de ICMPv6 para algunos de sus servicios, a continuación se definen los tipos del 133 al 137.

- **Valores de Tipo y Código de Descubrimiento de vecinos ICMPv6**, estas constantes están definidas en <netinet/icmp6.h>.

```
#define ND_ROUTER_SOLICIT        133
#define ND_ROUTER_ADVERT         134
#define ND_NEIGHBOR_SOLICIT      135
#define ND_NEIGHBOR_ADVERT       136
#define ND_REDIRECT              137

struct nd_router_solicit {     /* Solicitud de ruteador */
 struct icmp6_hdr  nd_rs_hdr;
  /* puede ser seguido por opciones */
};

struct nd_router_advert {      /* Anunciación del ruteador */
 struct icmp6_hdr  nd_ra_hdr;
 uint32_t   nd_ra_reachable;  /* tiempo accesible */
 uint32_t   nd_ra_retransmit; /* retransmitir cronómetro */
  /* puede ser seguido por opciones */
};

struct nd_neighbor_solicit {   /* Solicitud de vecino */
 struct icmp6_hdr  nd_ns_hdr;
 struct in6_addr   nd_ns_target; /* dirección objetivo */
  /* puede ser seguido por opciones */
};

struct nd_neighbor_advert {    /* anunciación de vecino */
 struct icmp6_hdr  nd_na_hdr;
 struct in6_addr   nd_na_target; /* dirección objetivo */
  /* puede ser seguido por opciones */
};

struct nd_redirect {           /* redirección */
 struct icmp6_hdr  nd_rd_hdr;
 struct in6_addr   nd_rd_target; /* dirección objetivo */
 struct in6_addr   nd_rd_dst;    /* dirección destino */
  /* could be followed by options */

};
```

```
struct nd_opt_hdr {          /* Opción de encabezado descubrimiento
                                                de vecino */
 uint8_t  nd_opt_type;
 uint8_t  nd_opt_len;        /* en unidades de 8 bytes */
  /* puede ser seguido por opciones */
};

struct nd_opt_prefix_info {   /* información de prefijo */
 uint8_t  nd_opt_pi_type;
 uint8_t  nd_opt_pi_len;
 uint8_t  nd_opt_pi_prefix_len;
 uint8_t  nd_opt_pi_flags_reserved;
 uint32_t  nd_opt_pi_valid_time;
 uint32_t  nd_opt_pi_preferred_time;
 uint32_t  nd_opt_pi_reserved2;
 struct in6_addr  nd_opt_pi_prefix;
};

struct nd_opt_rd_hdr {        /* encabezado redireccionado  */
 uint8_t  nd_opt_rh_type;
 uint8_t  nd_opt_rh_len;
 uint16_t  nd_opt_rh_reserved1;
 uint32_t  nd_opt_rh_reserved2;
  /* followed by IP header and data */
};

struct nd_opt_mtu {           /* opción MTU  */
 uint8_t  nd_opt_mtu_type;
 uint8_t  nd_opt_mtu_len;
 uint16_t  nd_opt_mtu_reserved;
 uint32_t  nd_opt_mtu_mtu;
};
```

Como ya se ha mencionado, AVICAR para IPv4 cuenta con tres módulos encargados de capturar paquetes ICMP, ARP y RARP. Al hacer los cambios para IPv6, sólo será necesario un módulo, el cual tendrá como función capturar los paquetes IP que contengan como encabezado de extensión un paquete ICMP. En la parte encargada de dibujar la interfaz gráfica sólo será necesario ajustar los mensajes que indican el tipo de mensaje que representa cada barra de la gráfica.

CONCLUSIONES

- AVICAR a diferencia de las herramientas que fueron probadas, presenta la información tanto en modo texto como en modo gráfico, además dichas herramientas no presentan una clasificación detallada del tipo de paquetes escuchados.
- Las gráficas que realiza AVICAR permiten al administrador de red visualizar de manera rápida los porcentajes y cantidades de paquetes ICMP, ARP y RARP.
- AVICAR permite al administrador de red realizar un análisis posterior del tráfico capturado, al brindar la facilidad de guardarlo en archivos.
- Al construir gráficas sobre los tipos de paquetes capturados, AVICAR ayuda al administrador a detectar ataques de negación de servicios, o en el mejor de los casos, problemas en el funcionamiento de su red.
- AVICAR es una herramienta de código abierto que permitirá futuras modificaciones, entre ellas, la construcción de otros tipos de gráficas e implementación para redes IPv6, además se puede incorporar a un sistema experto para resolver problemas de red.
- Para que AVICAR funcione en redes IPv6 sería necesario reducir los tres módulos encargados de captura de paquetes a uno solo, dado que el protocolo ND hace uso de los mensajes ICMPv6.
- Los principales cambios a considerar para la implementación de AVICAR en IPv6 son las funciones utilizadas para establecer la comunicación con la interfaz de red y las estructuras de datos que almacenan los encabezados de los datagramas.

REFERENCIAS

RFC's

[1] Postel, J., *"Internet Protocol - DARPA Internet Program Protocol Specification"*, RFC 791, USC/Information Sciences Institute, Septiembre 1981.

[2] Postel, J., *"Internet Control Message Protocol"*, RFC 792, ISI, Septiembre 1981.

[3] Plummer, David C., *"An Ethernet Address Resolution Protocol or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware"*, RFC 826, Noviembre 1982.

[4] Finlayson, Mann & Mogul, Theimer, *"A Reverse Address Resolution Protocol"*, RFC 903, Stanford University, Junio 1984.

[5] Mogul, J., Postel, J., *"Internet Standard Subnetting Procedure"*, RFC 950, Agosto 1985.

[6] Braden, R., *"Requirements for Internet Hosts -- Communication Layers"*, RFC 1122, Internet Engineering Task Force, Octubre 1989.

[7] Deering, S., *"ICMP Router Discovery Messages"*, RFC 1256, Xerox PARC. Septiembre 1991.

[8] Huitema, C., & Chair, *"IAB Recommendation for an Intermediate Strategy to Address the Issue of Scaling"*, RFC 1481, Internet Architecture Board, Julio 1993.

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

[10] Rekhter, Y., & T. Li, *"An Architecture for IP Address Allocation with CIDR"*, RFC 1518, T.J. Watson Research Center, IBM Corp., Cisco Systems, Septiembre 1993.

[11] Fuller, V., Li, T., Yu, J., & K. Varadhan, *"Classless Inter-Domain Routing (CIDR): An Address Assignment and Aggregation Strategy"*, RFC 1519, BARRNet, Cisco, MERIT, OARnet, Septiembre 1993.

[12] Baker, F., *"Requirements for IP Version 4 Routers"*, RFC 1812, Cisco Systems, Junio 1995.

[13] Stevens, W., & Thomas, M., *"Advanced Sockets API for IPv6"*, RFC 2292, AltaVista, Febrero 1998.

[14] Hinden, R., & Deering, S., *"IP Version 6 Addressing Architecture"*, RFC 2373, Nokia, Cisco Systems, Julio 1998.

[15] Deering, S., & Hinden, R., *"Internet Protocol, Version 6 (IPv6)"*, RFC 2460, Cisco, Nokia, Diciembre 1998.

[16] Narten, T., Nordmark, E., & Simpson, W., *"Neighbor Discovery for IP Version 6 (IPv6)"*, RFC 2461, IBM, Sun Microsystems, Daydreamer, Diciembre 1998.

[17] Conta, A., & Deering, S., *"Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6)"*, RFC 2463, Lucent, Cisco Systems, Diciembre 1998.

[18] Gilligan, R., Thomson, S., Bound, J., & Stevens, W., *"Basic Socket Interface Extensions for IPv6"*, RFC 2553, FreeGate, Bellcore, Compaq, Consultant, Marzo 1999.

[19] Deering, S., Fenner, W., & Haberman, B., *"Multicast Listener Discovery (MLD) for IPv6"*, RFC 2710, Cisco Systems, AT&T Research, IBM, Octubre 1999.

[20] Crawford, M., *"Router Renumbering for IPv6"*, RFC 2894, Fermilab, Agosto 2000.

[21] Conta, A., *"Extensions to IPv6 Neighbor Discovery for Inverse Discovery"*, RFC 3122, Transwitch Corporation, Junio 2001.

Páginas Web

[22] Arkin, Ofir *"ICMP Usage in Scanning -The Complete Know-How"*, Versión 3.0 Junio 2001, *http://www.sys-security.com/html/papers.html*

[23] *Echok, Smurf, Smurf5*, http://www.cotse.com/dos.htm

[24] *Tcpdump*, http://www.tcpdump.org/

[25] *Exdump*, http://packetstormsecurity.org/sniffers/exdump/

[26] *Snort*, http://archives.neohapsis.com/archives/snort/2000-07/0073.html

[27] *"Ataques de Negación de Servicios"*, http://www.unam-cert.unam.mx/negacion.html

[28] *"Flood un clásico"*, http://members.nbci.com/minos007/Minos/flood.htm

[29] Kirch Olaf, Dawson Terry, *"Linux Network Administrators Guide"*, http://www.linuxdoc.org/LDP/nag2/x-082-2-firewall.attacks.html

[30] Fenzi Kevin, Dave Wreski, *"Linux Security HOWTO"*, traducido por Pedro Pablo Fábrega, http://www.arrakis.es/~pfabrega/seguridad-como.html

[31] Etcheverry Javier, Porro Sebastian, *"Seguridad en Internet (Parte 2)"*, http://www.nuia.com.ar/nuia44/SEGURIDA.HTM

[32] Gómez Skarmeta Antonio, Ibáñez Martínez Jesús, Martínez Barberá Humberto, *"Seguridad en Internet: Ataques, Técnicas y Firewalls"*, http://ants.dif.um.es/~humberto/pub/Novatica97-1/Novatica97-1.html

[33] Palet Martínez, Jordi, *"Tutorial de IPv6"*, http://www.consulintel.es/Html/ForoIPv6/Documentos.htm

[34] *"IPV6 – Internet"*, http://info.telecomco.net/unidadtrans/gruposint/internet/temas/ipv6.htm

[35] Millán Tejedor, Ramón Jesús, *"EL PROTOCOLO IPV6 (Y II) NUEVA ESTRUCTURA DE DIRECCIONES EN INTERNET"*, http://www.linuxworld.com/english/crd_ipv6_529707.html

[36] Network Sorcery, Inc., *"ICMPV6, INTERNET CONTROL MESSAGE PROTOCOL FOR IPV6"*, http://www.networksorcery.com/enp/protocol/icmpv6.htm

Libros

[37] Comer, Douglas E. *"REDES GLOBALES DE INFORMACIÓN CON INTERNET Y TCP/IP, Principios básicos, protocolos y arquitectura"*, Tercera Edición, Prentice Hall, 1996.

[38] Foster, Eric, Johnson, *"Graphical Applications With Tcl & Tk"*, Segunda Edición, M&T Books, 1997.

[39] Márquez García Francisco Manuel, *"Unix Programación Avanzada"*, Addison-Wesley Iberoamericana, RA-MA 1993.

[40] Miller, Stewart S., *"The Next Generation Internet Protocol"*, Digital Press, 1997.

[41] Naugle, Matthew G. *"Network Protocols"*, Signature Edition, McGraw-Hill, 1998.

[42] Schildt Herbert, *"Turbo C/C++, Manual de Referencia"*, Mc Graw Hill, 1992.

[43] Stevens, W. Richard, *"TCP/IP Illustrated Vol. 1, The Protocols"*, Addison-Wesley Professional Computing Series, 1994.

[44] Stevens, W. Richard., *"Unix Network Programming"*, Prentice Hall, 1990.

Artículo

[45] Trejo, Luis A., *"Ataques Activos más Comunes en IP"*, Soluciones Avanzadas Tecnologías de Información y Estrategias de Negocios, Año 5, Número 44, 15 de Abril 97.

Apéndice A

MANUAL DEL USUARIO

En este apéndice se explica el funcionamiento de AVICAR (Analizador y Visualizador de paquetes ICmp, Arp y Rarp). Ésta es una herramienta que permite al usuario (administrador de red) capturar y visualizar tanto los paquetes ICMP, ARP y RARP como las gráficas del número de paquetes recibidos de cada tipo.

AVICAR

Para utilizar esta herramienta es necesario ser el superusuario de una máquina conectada a una red de área local que trabaje con el sistema operativo Linux y que tenga instalado el intérprete de Tk (wish8.0), que en Linux normalmente es llamado simplemente *wish*. En caso de no contar con el intérprete se puede obtener de Internet.

Avicar está formado por los siguientes módulos:

- Tres módulos que escuchan paquetes: sniffer para paquetes ICMP, sniffer2 para paquetes ARP y sniffer3 para paquetes RARP,
- El módulo que presenta el menú principal de la interfaz gráfica, y
- Tres módulos que presentan los datos y gráficas de paquetes ICMP, ARP y RARP.

Los tres módulos que escuchan paquetes son manipulados por los módulos que construyen la interfaz gráfica. Por lo que se va a describir la forma de uso de estos últimos solamente.

Para poner en funcionamiento a Avicar, se requiere descomprimir el archivo **avicar_v1.0.tgz**, aplicando el siguiente comando:

tar –xvzf avicar_v1.0.tgz

éste va a crear una carpeta llamada Avicar en la ruta donde se haya aplicado el comando, por lo que el usuario tendrá que moverse a esta carpeta y desde línea de comandos ejecutar ./Avicar.tcl.

MENÚ PRINCIPAL

Al momento de ejecutar AVICAR, se presenta el menú principal (figura A.1), el cual contiene tres botones de chequeo y un botón, al dar un click sobre alguno de los botones de chequeo se activa la herramienta encargada de escuchar, presentar y graficar los paquetes correspondientes, dependiendo la leyenda del botón de chequeo activado. El botón sirve para terminar con la ejecución de AVICAR.

Figura A.1. Menú principal de AVICAR

ESCUCHANDO PAQUETES ICMP

Al activar el botón de chequeo ICMP, el menú principal se verá como lo muestra la figura A.2, e inmediatamente después se desplegará la pantalla mostrada en la figura A.3.

Figura A.2. Menú principal con botón de chequeo ICMP activado

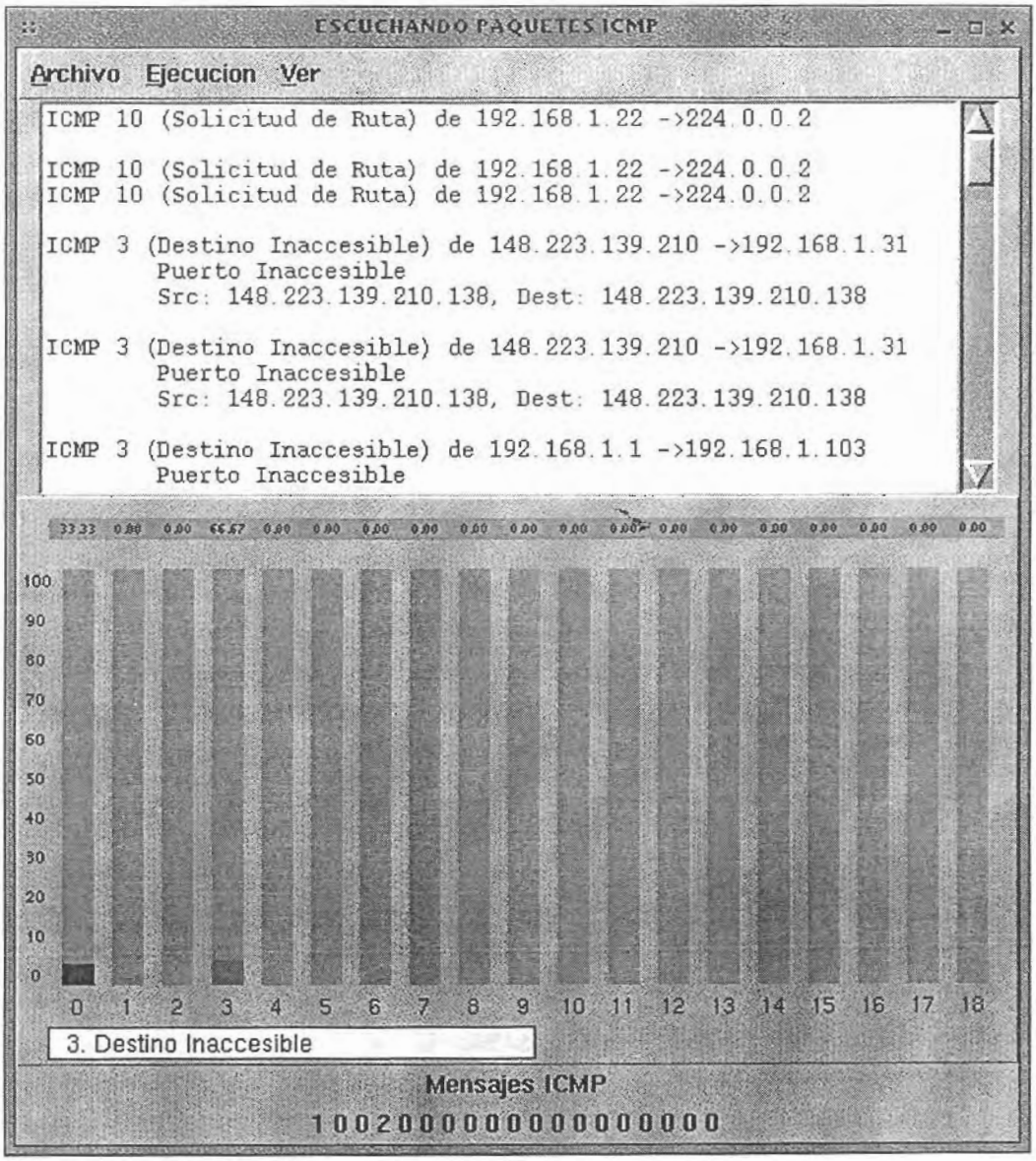


Figura A.3. Escuchando paquetes ICMP

Esta pantalla despliega los paquetes que se van leyendo del archivo generado por el sniffer (ICMP), así como la gráfica generada a partir de ellos. Estas gráficas representan el número de paquetes escuchados de cada tipo en un tiempo determinado, además de presentar en la parte superior de cada columna el porcentaje que representa cada una con respecto al número total de paquetes escuchados. Al pasar el mouse sobre alguna de las columnas se presenta en la parte inferior izquierda el tipo de mensaje que representa la columna. La gráfica se va actualizando periódicamente.

El menú que se presenta en esta pantalla en la parte superior tiene las siguientes funciones:

- Archivo
  - Abrir, permite abrir un archivo que haya sido guardado por medio de la misma herramienta.
  - Guardar, permite guardar en un archivo los paquetes escuchados durante un periodo de tiempo.
  - Salir, termina con la ejecución del módulo.

- Ejecución
  - Detener, deja de escuchar paquetes, se pueden usar las teclas rápidas <Ctrl+d>.
  - Continuar, continúa escuchando paquetes, se pueden usar las teclas rápidas <Ctrl-c>.

- Ver
  - ICMP, visualiza la gráfica general de los paquetes escuchados, su manera de presentarla se muestra en la figura A.4. Esta pantalla presenta un botón Ocultar, cuya función es, como su nombre lo dice, ocultar la gráfica general.

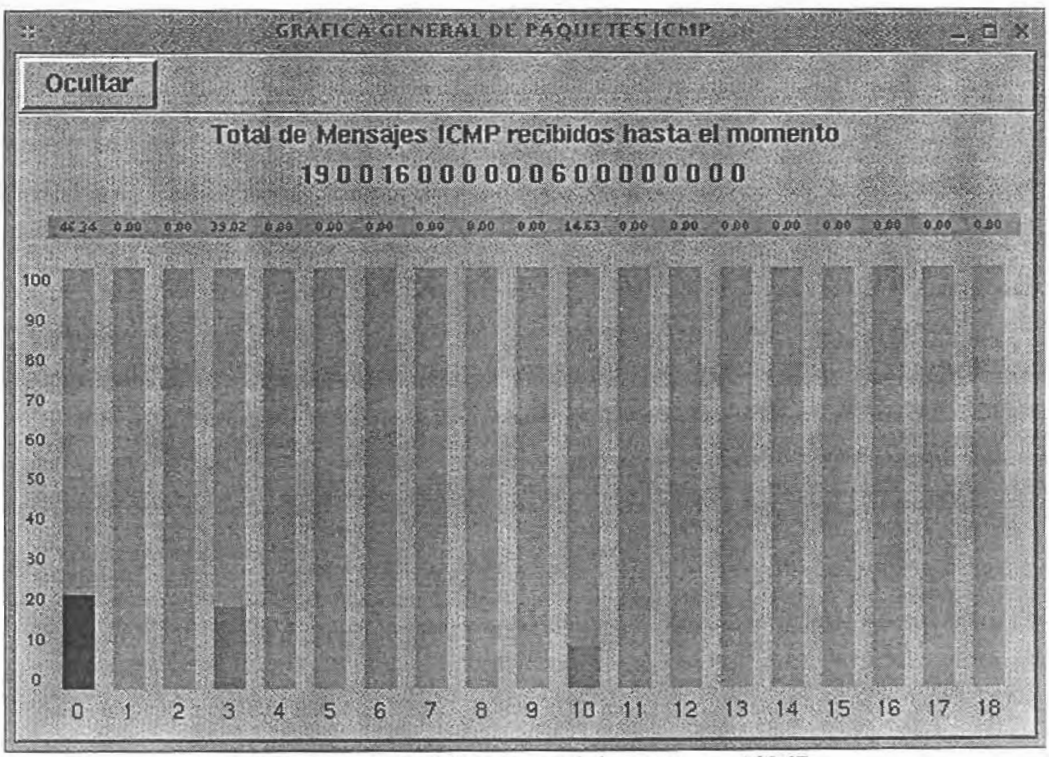


Figura A.4. Gráfica general de paquetes ICMP

ESCUCHANDO PAQUETES ARP

Al activar el botón de chequeo ARP, el menú principal se verá como lo muestra la figura A.5, e inmediatamente después se desplegará la pantalla mostrada en la figura A.6.

Figura A.5. Menú principal con botón de chequeo ARP activado

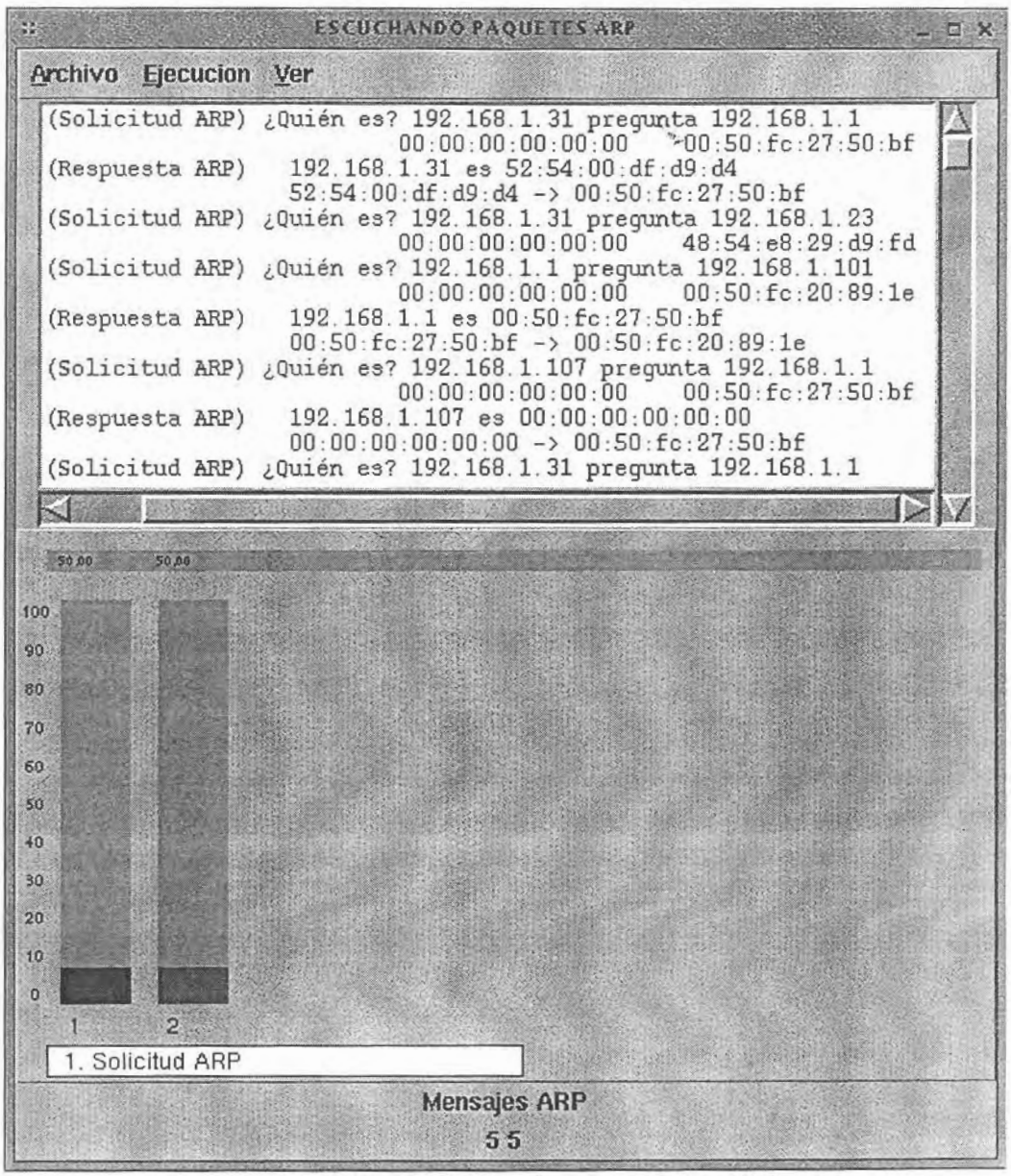


Figura A.6. Escuchando paquetes ARP

Esta pantalla despliega los paquetes que se van leyendo del archivo generado por el sniffer2 (ARP), así como la gráficas generada a partir de ellos. Tiene las mismas funciones implementadas para la pantalla de ICMP, con la diferencia de que en la opción Ver, despliega la gráfica general de paquetes ARP.

Al escuchar paquetes RARP, las pantallas presentadas son muy parecidas a las de ARP, con la diferencia del tipo de paquetes que escucha.

Si se quieren escuchar los tres tipos de paquetes al mismo tiempo es posible hacerlo.