PONTIFICIA UNIVERSIDAD CATOLICA DE VALPARAISO

Diego Franco Arancibia Uribe

Implementación de sniffer en plataforma de código abierto

Informe Proyecto de Título de Ingeniero Civil Electrónico

Escuela de Ingeniería Eléctrica

Implementación de sniffer en plataforma de código abierto

Diego Franco Arancibia Uribe

Tesis para el grado de Ingeniero Civil Electrónico,

aprobada por la comisión de la

Escuela de Ingeniería Eléctrica de la

Pontificia Universidad Católica de Valparaíso

conformada por

Sr. David Alberto Velasco López

Profesor Guía

Sr. Francisco Javier Alonso Villalobos

Segundo Revisor

Sr. Sebastián Fingerhuth Massmann

Secretario Académico

Valparaíso, 01 de Agosto de 2017

Resumen

Partiendo por un repaso a la historia y fundamentos del estándar IEEE 802.3, este documento trata sobre el desarrollo e implementación de un analizador de paquetes en un entorno de redes Ethernet.

Partiendo de lo más básico, el documento explica el modelo de interconexión de sistemas abiertos (OSI por sus siglas en inglés) y sus siete capas, de las cuales la capa de datos y de red son el ambiente en el cual se desarrolla el estándar Ethernet. Luego de esto, se hace el estudio de dicho estándar, además de los protocolos TCP/IP, los cuales forman el conjunto de reglas y procedimientos mediante los cuales los datos serán transmitidos en una red. Se hace un breve repaso de los principales protocolos a reconocer por el analizador de paquetes, comentando sus funciones y características fundamentales.

Antes de desarrollar este sniffer, se hace una comparativa de dos analizadores de paquetes existentes en el mercado, con el fin de tener una perspectiva clara, de cómo deben ser las características fundamentales de éste.

Finalmente, se presenta la estructura del proyecto, los dispositivos a utilizar en su realización y el desarrollo mediante programación del analizador de paquetes, para luego mostrar su funcionamiento.

Palabras claves: sniffer, protocolos de comunicación, redes de computadores, sistema operativo, ethernet, MAC, IP

Abstract

Starting with a review of history and fundamentals about IEEE 802.3 standard. This document is about the development and implementation of a sniffer in an Ethernet network.

First, the document explains the OSI model and its seven layers, of which the data and network layers, are the environment in which the standard Ethernet works. After this, the study of that standard is developed, also the TCP/IP protocols, which performs the conjunct of rules and procedures to transmit the data in a network. Then, a review of principal protocols to analyze by the sniffer are made, remarking its functions and characteristics.

Before developing this sniffer, a comparative is made between two sniffers existing in the market, in order to have a clear view about how must be the principal characteristics of this.

Finally, the structure project, devices to use, and programming development are presented, to show later its operation.

Key words: sniffer, communication protocols, computer network, operating system, Ethernet, MAC, IP

Índice general

Introducción 1

1 Analizador de paquetes y estandar Ethernet 3
1.1 Modelo de referencia OSI 4
1.1.1 Capa de aplicación 5
1.1.2 Capa de presentación 5
1.1.3 Capa de sesión 5
1.1.4 Capa de transporte 5
1.1.5 Capa de red 5
1.1.6 Capa de enlace de datos 5
1.1.7 Capa física 6
1.2 Encapsulamiento 6
1.3 Estandar Ethernet 8
1.4 Protocolos TCP/IP 8
1.4.1 Protocolo IP 9
1.4.2 Protocolo UPD 10
1.4.3 Protocolo TCP 11
1.4.4 Protocolo ICMP 12
1.4.5 Protocolo ARP 13
1.4.6 Protocolo DNS 14
1.5 Análisis de sniffers existentes 15
1.5.1 Acrylic 15
1.5.2 “Sniffing” con Acrylic 15
1.5.3 Análisis de los datos obtenidos con Acrylic 17
1.6 Wireshark 18
1.6.1 Introducción al software 18
1.7 Análisis de Acrylic y Wireshark 18
1.8 Problemática y abordaje de la solución 19
1.9 Objetivos 20
1.10 Solución propuesta 20

2 Estructura del proyecto 21

2.1 Controlador RTL8169 21
2.1.1 Registros RTL8169 21
2.2 Entorno de trabajo 23
2.3 Estructura del analizador de paquetes en DANUX 23
2.4 Inicialización de la tarjeta de red 24
2.5 Asignación de direcciones DMA 25
2.6 Habilitación de recepción de tramas 25

3 Desarrollo del Sniffer 28
3.1 Producción principal 28
3.1.1 Reconocimiento de protocolos 31
3.2 Planteamiento 31
3.3 Programación final del sniffer 32
3.3.1 Programa principal 32
3.3.2 Producciones 32
3.4 Reconocimiento trama IPv4 34
3.5 Reconocimiento de tramas en la capa de transporte 35
3.6 Reconocimiento trama ICMP 36
3.7 Reconocimiento trama ARP 37
3.8 Reconocimiento trama DNS 37
3.9 Reconocimiento de tramas extra 38

Discusión y conclusiones 39

Bibliografía 40

Introducción

Las redes están compuestas por diversos dispositivos que se desempeñan en conjunto para lograr un sistema funcional. Estos dispositivos de red son fabricados por diversas compañías, lo que genera la necesidad de que exista comunicación y entendimiento entre los fabricantes, con respecto a cómo operan e interactúan los distintos dispositivos de una red. Es por esto que se han creado estándares que definen la forma de conectar los dispositivos y los protocolos necesarios cuando se establecen comunicaciones.

Originalmente diseñado en la década de los setenta, el estándar Ethernet ha evolucionado de una red half-duplex de 10 Mbps, a una tecnología de switch full-duplex de 1000 Mbps. El sistema de red Ethernet fue originalmente creado por Xerox, pero desarrollado conjuntamente como una norma por *Digital Equipment Corporation*, Intel y Xerox, esta norma se conoció como DIX Ethernet, haciendo referencia a los nombres de quienes lo habían desarrollado.

Estandarizado por el comité 802.3 del instituto de ingenieros eléctricos y electrónicos (IEEE), la palabra Ethernet no es mencionada en el estándar. Sin embargo, es comúnmente referenciado como Ethernet y puede ser encontrado en cualquier parte, incluso en sistemas de automatización. El instituto de ingenieros eléctricos y electrónicos, en los años setenta, puso en funcionamiento un proyecto llamado *comité 802,* esto con el fin de estandarizar una tecnología que permitiera incorporar los diferentes productos comerciales disponibles en redes locales hasta el momento. De este modo, se lograría establecer una libre competencia, bajar los precios y así beneficiar a los usuarios. Tiempo después de haberse dado a conocer este comité, DIX le informó que Ethernet ya estaba en su etapa final, y que cuando estuviera terminado lo propondría para estandarización. Finalmente, IEEE aceptó la propuesta y creó el subcomité 802.3 para Ethernet.

Ethernet sólo define las dos capas (física y de datos) más bajas del modelo OSI (Open Systems Interconnection), pero TCP/IP soporta las capas de transporte y red. Utiliza un método de acceso sensible a la señal portadora, mediante el cual las estaciones de trabajo comparten un cable de red, pero sólo una de ellas puede utilizarlo en un momento dado. El método de Acceso Múltiple con Detección de Portadora y Detección de colisiones (CSMA/CD, Carrier Sense Multiple Access with Collision Detection), se utiliza para arbitrar el cable. CSMA/CD incorpora dos características fundamentales: no se transmite si hay otra estación *hablando*, y segundo, si mientras se está transmitiendo, detecta que otra estación transmite (se produce una colisión), la estación deja de transmitir.

El conjunto de protocolos TCP/IP se utilizan para establecer comunicación entre diferentes nodos en un entorno heterogéneo y definen los formatos y normas utilizados en la transmisión y recepción de información [1]. En el conjunto de protocolos TCP/IP, todos los paquetes se entregan mediante el servicio de entrega de datagramas IP. IP define el formato que los paquetes deben tener y el modo de utilizarlos durante el envío y la recepción.

Según IDC (International Data Corporation), a finales del año 1997, más del 85% de las conexiones de red instaladas en el mundo eran Ethernet. Así, la hegemonía de las redes locales Ethernet desde su comienzo solo ha ido en aumento.

Hoy en día Ethernet se ocupa tanto a nivel domiciliario, como a nivel industrial. Protocolos tales como Modbus TCP, EtherCAT, Ethernet/IP, PROFINET, hacen que el estándar sea de uso habitual en redes industriales. Esto implica la necesidad de una herramienta que sea capaz de analizar estos protocolos y que permita el análisis y mantenimiento de estas redes. En la actualidad dicha herramienta es conocida como analizador de paquetes o *sniffer* (olfatear) en inglés. Un sniffer es un programa que trabaja en conjunto con la tarjeta de interfaz de red, para obtener todo el tráfico que esté dentro de la red, y no sólo el tráfico dirigido a dicha tarjeta, sino a todas las direcciones presentes.

Este proyecto se basa en la creación de un dispositivo que sea capaz de analizar los principales protocolos de comunicación basados en el estándar Ethernet, sin ser detectado por otros dispositivos pertenecientes a la red. Si bien los analizadores de paquetes ya existen en el mercado, no existe alguno que sea indetectable a los demás usuarios pertenecientes a una red Ethernet. Debido a la infinidad de procesos de enviar y recibir datos, dados los sistemas operativos existentes, las tarjetas de red están constantemente enviando información, lo cual hace al computador automáticamente detectable en la red, además de ralentizar el proceso de obtención de paquetes.

El objetivo general del proyecto es lograr un sniffer indetectable, el cual solo recibirá paquetes, sin enviar información de vuelta, además de lograr una gran velocidad de captura de tramas, que impida la perdida de información.

1 Analizador de paquetes y estandar Ethernet

Un sniffer o analizador de paquetes, es un programa o dispositivo que captura las tramas de una red de computadoras.

Es común que el medio de transmisión en una red, sea compartido por varias computadoras y dispositivos de red, lo que hace posible que un ordenador pueda capturar las tramas no destinadas a él. Este proceso se realiza mediante un sniffer, el cual le ordena a la computadora a que capture todo el tráfico de datos, incluyendo las tramas que no están destinadas al equipo, esto se conoce como poner en estado “promiscuo” a la tarjeta de red inalámbrica. Una vez que la tarjeta de red se encuentra en este estado, el programa comienza a hacer una lectura de toda la información entrante a la computadora, con esto el sniffer conseguirá obtener tanto la información enviada, como los datos correspondientes a los dispositivos emisor y receptor (Dirección MAC y dirección IP).

El uso que se le da a este tipo de aplicaciones es algo importante de señalar, ya que pueden ayudar a tener mayor seguridad en una red, hacer pruebas, diagnósticos, etc. Los principales usos que se le pueden dar a un sniffer son:

- Captura de nombres de usuario y contraseñas enviadas sin cifrar en una red.
- Análisis de fallos en la red, tales como ¿Por qué un ordenador no puede establecer comunicación con otro?
- Medición de tráfico, para identificar atascamiento de tramas.
- Desarrollo de aplicaciones y protocolos cliente-servidor. Permite analizar la información real que se transmite por la red.

Algunos de los sniffer más utilizados en la actualidad son:

- Wireshark: Utilizado para efectuar análisis y solucionar problemas en redes para desarrollo de software y protocolos, y como una herramienta didáctica para educación. Cuenta con todas las características estándar de un analizador de protocolos [2].
- Acrylic: Similar a Wireshark, pero con distintas versiones pagadas, las cuales varían en su precio, dependiendo de las prestaciones que tengan [3].

- Ettercap: Soporta direcciones activas y pasivas de varios protocolos. También hace posible la inyección de datos en una conexión establecida. Contiene varios modos de "sniffing", lo que convierte a este software en una herramienta completa de análisis [4].
- Kismet: Es un sniffer y además un sistema de detección de intrusiones para redes inalámbricas 802.11 [5].

1.1 Modelo de referencia OSI

El modelo de referencia OSI, es el modelo principal para las comunicaciones por red. Aunque existen otros modelos, en la actualidad la mayoría de los fabricantes de redes relacionan sus productos con el modelo OSI.

En el modelo de referencia OSI existen siete capas, las cuales indican una función de red específica, esto sirve para comprender cómo viaja la información a través de una red. El dividir la red en estas siete capas genera las siguientes ventajas:

- Divide la comunicación de red en partes más pequeñas y sencillas
- Normaliza los componentes de red para permitir el desarrollo y el soporte de los productos de diferentes fabricantes
- Permite a los distintos tipos de hardware y software de red comunicarse entre sí.
- Impide que los cambios en una capa puedan afectar a las demás.

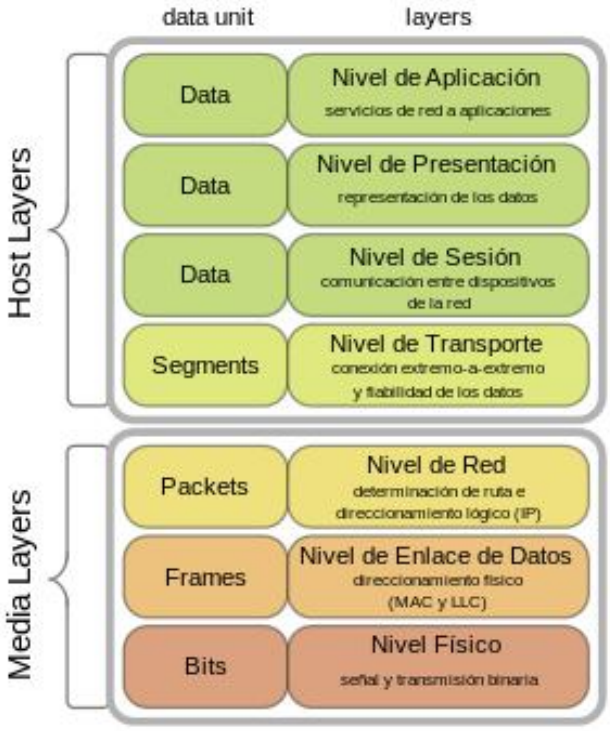

Figura 1-1 Modelo OSI (fuente: es.pinterest.com).

La Figura 1-1 muestra la estructura del modelo OSI, cada capa individual del modelo OSI tiene un conjunto de funciones que debe realizar para que la información pueda viajar en la red desde el origen hasta el destino.

1.1.1 Capa de aplicación

La capa de aplicación es la capa más cercana al usuario, suministra servicios de red a las aplicaciones del usuario, y difiere de las demás capas debido a que no proporciona servicios a ninguna otra capa OSI, sino que solamente a aplicaciones que se encuentran fuera del modelo OSI [6]. Un ejemplo de aplicación puede ser un programa de hojas de cálculo. La capa de aplicación establece la disponibilidad de los potenciales socios de comunicación, sincroniza y establece acuerdos sobre los procedimientos de recuperación de errores y control de la integridad de los datos.

1.1.2 Capa de presentación

La capa de presentación garantiza que la información que envía la capa de aplicación de un sistema, pueda ser leída por la capa de aplicación de otro [6].

1.1.3 Capa de sesión

La capa de sesión establece, administra y finaliza las sesiones entre dos hosts que se estén comunicando. Proporciona sus servicios a la capa de presentación. También sincroniza el diálogo entre las capas de presentación de los dos hosts y administra su intercambio de datos.

1.1.4 Capa de transporte

La capa de transporte segmenta los datos originados en el host emisor y los reensambla en una corriente de datos, dentro del sistema del host receptor.

Esta capa suministra un servicio de transporte de datos que aísla las capas superiores de los detalles de implementación del transporte. Al proporcionar un servicio de comunicación, establece, mantiene y termina adecuadamente los circuitos virtuales. En ella se utilizan dispositivos de detección y recuperación de errores de transporte.

1.1.5 Capa de red

La capa de red proporciona conectividad y selección de ruta entre dos sistemas de hosts que pueden estar ubicados en redes geográficamente distintas.

1.1.6 Capa de enlace de datos

La capa de enlace de datos, proporciona tránsito de datos confiable a través de un enlace físico. Se ocupa del direccionamiento físico, topología de la red, acceso a la red, notificación de errores, entrega ordenada de tramas y control de flujo.

1.1.7 Capa física

La capa física define las especificaciones eléctricas, mecánicas, de procedimiento y funcionales para activar, mantener y desactivar el enlace físico entre sistemas finales. Las características tales como niveles de voltaje, temporización, velocidad de datos físicos, distancias de transmisión máximas, conectores físicos y otros atributos similares, son definidos por las especificaciones de la capa física.

1.2 Encapsulamiento

Todas las comunicaciones de una red, parten de un origen y se envían a un destino. Si un ordenador (host A) desea enviar datos a otro (host B), los datos deben ser previamente empaquetados mediante el proceso denominado encapsulamiento.

El encapsulamiento rodea los datos con la información de protocolo necesaria, antes de que se una al tránsito de la red. Por lo tanto, a medida que los datos se desplazan a través de las capas del modelo OSI, reciben encabezados, información final y otros tipos de información. El proceso se ilustra en la Figura 1-2.

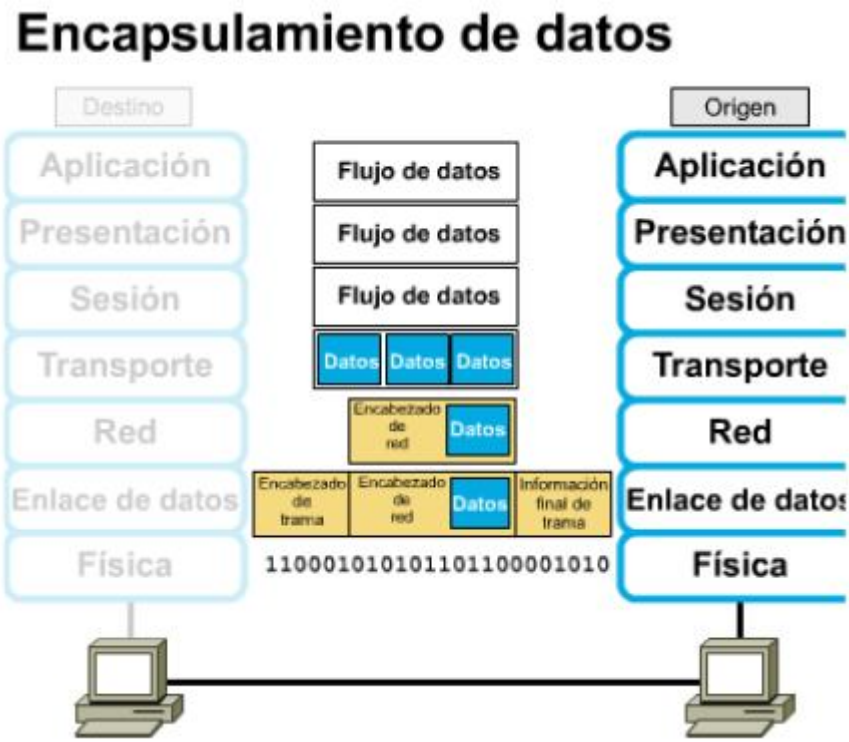

Figura 1-2 Encapsulamiento de datos (fuente: www.mastermagazine.info).

Una vez que se envían los datos desde el origen, estos viajan a través de la capa de aplicación y recorren todas las demás capas en orden descendente. El empaquetamiento y el flujo de los datos que se intercambian experimentan cambios a medida que las redes ofrecen sus servicios a los

usuarios finales. Como muestra la Figura 1-3, las redes deben realizar cinco pasos de conversión, a fin de encapsular los datos.

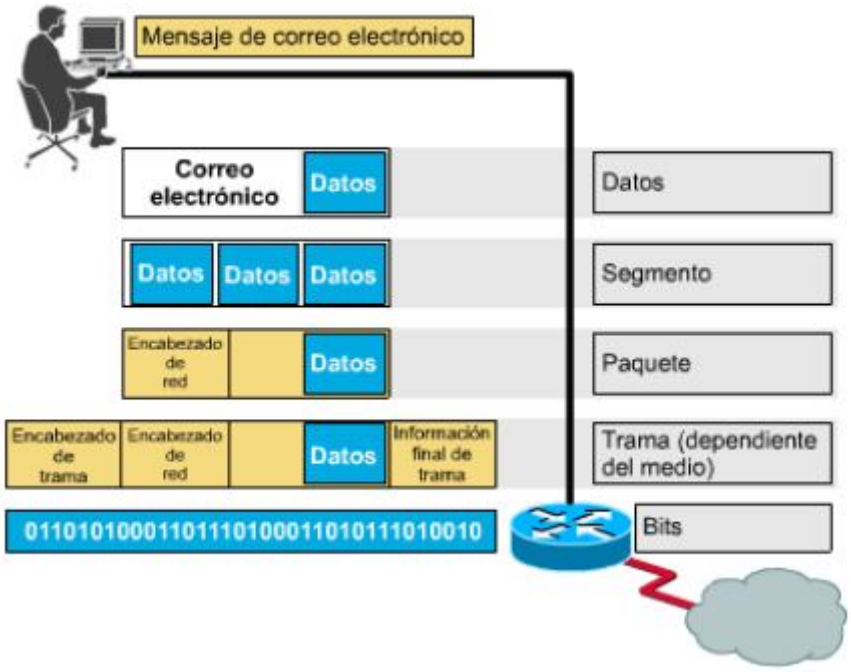

Figura 1-3 Ejemplo de encapsulamiento de datos (fuente: classunaed.blogspot.cl).

1. Creación de datos: Cuando un usuario envía un mensaje de correo electrónico, sus caracteres alfanuméricos se convierten en datos.
2. Empaquetar datos para ser transportados: Al utilizar segmentos, la función de transporte asegura que los hosts del mensaje en ambos extremos del sistema de correo electrónico se puedan comunicar de forma confiable.
3. Agregar la dirección de red al encabezado: Los datos se colocan en un paquete o datagrama que contiene el encabezado de red con las direcciones lógicas de origen y de destino. Estas direcciones ayudan a los dispositivos de red a enviar los paquetes a través de la red por una ruta seleccionada.
4. Agregar la dirección local al encabezado de enlace de datos: Cada dispositivo de la red debe poner el paquete dentro de una trama. La trama le permite conectarse al próximo dispositivo de red conectado directamente en el enlace. Cada dispositivo en la ruta de red seleccionada, requiere el entramado para poder conectarse al siguiente dispositivo.
5. Realizar la conversión a bits para su transmisión: La trama debe convertirse en un patrón de unos y ceros (bits) para su transmisión a través del medio. Una función de temporización permite que los dispositivos distingan estos bits a medida que se trasladan por el medio.

1.3 Estandar Ethernet

También conocido como IEEE 802.3, es el estándar de redes de área local más conocido, usa el método de transmisión de datos llamado "Acceso múltiple con detección de portadora y detección de colisiones (CSMA/CD)" [1]. Antes de que un nodo envíe algún dato a través de una red Ethernet, primero escucha y se da cuenta si algún otro nodo está transfiriendo información, de no ser así, el nodo transferirá la información a través de la red. Todos los otros nodos escucharán y el nodo seleccionado recibirá la información. Cada estación recibe una copia de cada paquete, pero ignora los que son dirigidos a otros computadores y procesa solamente los que son dirigidos a ella, este punto es muy importante, dado que, para que el sniffer funcione de manera efectiva, se debe tener la tarjeta de red en modo promiscuo, con el fin de capturar toda trama que pase por delante.

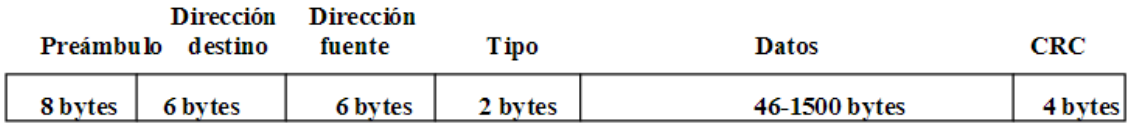

Figura 1-4 Trama Ethernet (fuente: propia).

En la Figura 1-4 se aprecia la trama Ethernet genérica, cabe destacar que en las capturas no se muestra el campo "Preámbulo", ya que el hardware de la tarjeta de red lo elimina, pues no forma parte propiamente de la trama Ethernet. Lo mismo sucede con el campo CRC (Cyclic Redundancy Check), el hardware detecta que es correcto y lo elimina.

El campo "Tipo", contiene un entero de 16 bits que identifica el tipo de dato que es transferido en la trama. Por ejemplo, si se está transmitiendo un datagrama IP versión 4, el campo "Tipo", contendrá el valor hexadecimal "0800", si es de tipo IPv6, el campo tendrá el valor hexadecimal "86DD".

1.4 Protocolos TCP/IP

La familia de protocolos TCP/IP tiene correspondencia con el modelo de interconexión de sistemas abiertos (OSI por sus siglas en ingles). El modelo OSI describe un sistema ideal de redes que permite establecer una comunicación entre procesos de capas distintas. En el host, las capas prestan servicios a capas superiores y reciben servicios de las capas inferiores. La Figura 1-5 muestra las siete capas del modelo OSI y su correspondencia general con las capas del conjunto de protocolos Ethernet.

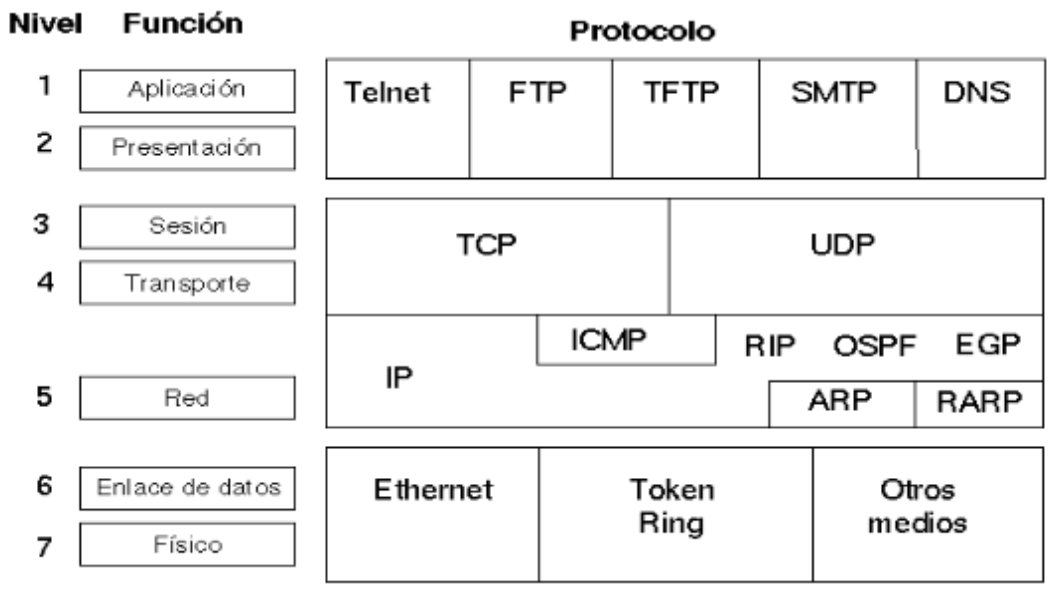

Figura 1-5 Modelo OSI y capas de TCP/IP (fuente: http://mixteco.utm.mx).

La Tabla 1-1 muestra los protocolos más comunes de TCP/IP

Tabla 1-1 Protocolos más comunes de TCP/IP

Protocolo	Servicio
IP	Proporciona servicios para la entrega de paquetes entre nodos
ICMP	Regula transmisión de mensajes de error y control entre los hosts y routers
ARP	Asigna direcciones de internet, a direcciones físicas
TCP	Proporciona servicios de envío de flujos fiables entre los clientes
UDP	Proporciona servicio de entrega de datagramas no fiable entre clientes
FTP	Proporciona servicios de nivel de aplicación para la transferencia de archivos

1.4.1 Protocolo IP

IP define el formato que los paquetes deben tener y el modo de utilizarlos durante el envío y la recepción. El formato que toma el paquete se denomina datagrama IP. La Figura 1-6 muestra una trama IP genérica.

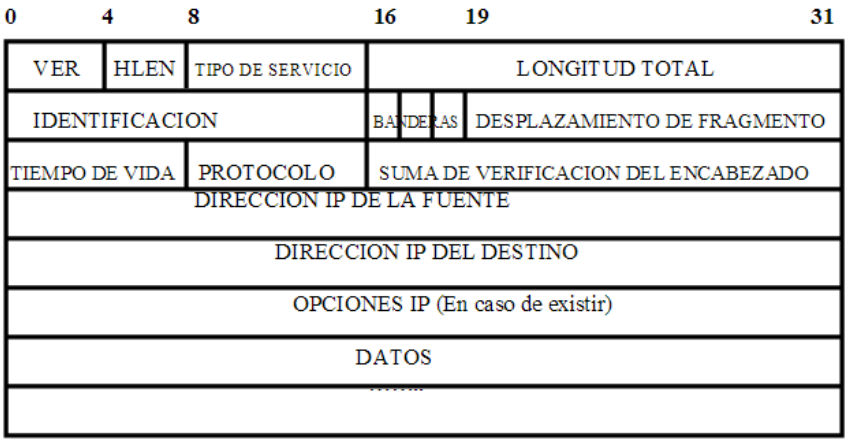

Figura 1-6 Datagrama IPv4 (fuente: http://mixteco.utm.mx).

El tamaño normal de la cabecera IP es de 20 bytes, a menos que presente el campo de "OPCIONES IP". Los campos del datagrama IP se describen a continuación: El campo VER es de 4 bits y contiene la versión del protocolo IP que se utilizó para crear el datagrama. HLEN es un campo de 4 bits, que proporciona la longitud del encabezado del datagrama medida en palabras de 32 bits. El encabezado común, que no contiene opciones ni rellenos, mide 20 bytes y tiene un campo de longitud igual a 5. El TIPO DESERVICIO es un campo de 8 bits que está subdividido en 5 campos, tres bits para especificar la prioridad del datagrama, los siguientes tres D, T y R especifican el tipo de transporte deseado para el datagrama, y los dos últimos no se utilizan. El campo LONGITUD TOTAL proporciona la longitud del datagrama medido en bytes, incluyendo los bytes del encabezado y los datos. El campo IDENTIFICACION contiene un entero único para identificar el datagrama. BANDERAS es un campo de tres bits que controlan la fragmentación, el primer bit no se utiliza, y el segundo es llamado DF que quiere decir no fragmentación y el tercero MF que significa más fragmentos. El campo DESPLAZAMIENTO DE FRAGMENTO especifica el desplazamiento en el datagrama original de los datos que se están acarreando en el fragmento. El campo TIEMPO DE VIDA, especifica la duración en segundos del tiempo que el datagrama tiene permitido permanecer en la red. El campo PROTOLOCO contiene un valor que especifica qué protocolo se utilizó para crear el mensaje que se está transportando en el área de datos. El campo SUMA DE VERIFICACIÓN DEL ENCABEZADO asegura la integridad de los valores del encabezado. Los campos DIRECCION IP DE LA FUENTE y DIRECCION IP DEL DESTINO contienen la dirección IP del emisor y del receptor respectivamente. El campo OPCIONES se incluye en principio para pruebas de red o depuración.

1.4.2 Protocolo UPD

Protocolo de datagrama de usuario, proporciona un servicio de entrega de datagramas no fiable entre clientes. La Figura 1-7 muestra la estructura básica del protocolo.

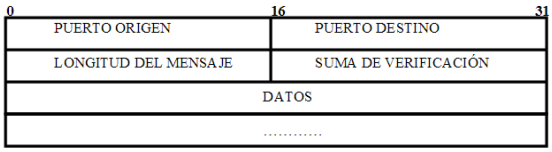

Figura 1-7 Datagrama UDP (fuente: http://mixteco.utm.mx).

1.4.3 Protocolo TCP

El "protocolo de control de transmisión" está pensado para ser utilizado como un protocolo 'host' a 'host' muy fiable entre miembros de redes de comunicación de computadoras por intercambio de paquetes y en un sistema interconectado de tales redes [7]. Ubicado en la capa de transporte del modelo OSI, es uno de los protocolos fundamentales en internet.

Las aplicaciones pueden comunicarse en forma segura mediante el protocolo TCP, debido al acuse de recibo (ACK) implementado en este.

TCP conecta un encabezado a los datos transmitidos, el cual contiene varios parámetros que ayudan a los procesos del sistema transmisor a conectarse a sus procesos correspondientes en el sistema receptor. El protocolo confirma que un paquete ha alcanzado su destino estableciendo una conexión de punto a punto entre los hosts de envío y recepción. Por tanto, el protocolo TCP se considera un protocolo fiable orientado a la conexión. El emisor envía un paquete e inicia un temporizador. La cabecera de cualquier paquete TCP contiene el número de secuencia del primer octeto de los datos en el segmento, cuando el receptor recibe el paquete, el TCP del receptor devuelve otro con el número de ACK igual al siguiente número de secuencia que espera recibir. Si el temporizador del emisor expira antes de la recepción del ACK, el transmisor envía el segmento de nuevo [1]. Esto garantiza la entrega fiable y ordenada de datos y fuerza un control de flujo entre emisor y receptor. Por otra parte, existen complicaciones tales como la fragmentación, pérdidas de ACK, duplicación de paquetes [8].

Offsets Octeto		0								1								2								3							
Octeto	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Puerto de origen																Puerto de destino															
4	32	Número de secuencia																															
8	64	Número de acuse de recibo (si ACK es establecido)																															
12	96	Longitud de Cabecera				Reservado				NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Tamaño de Ventana														
16	128	Suma de verificación																Puntero urgente (si URG es establecido)															
20 ...	160 ...	Opciones (Si la Longitud de Cabecera > 5, relleno al final con "0" bytes si es necesario) ...																															

Figura 1-8 Trama TCP (fuente: redesdecomunicacionindustrial.blogspot.cl).

La Figura 1-8 muestra la cabecera de una trama TCP genérica.

- Número de secuencia: Cuando una conexión está siendo establecida, el bit "SYN" se activa y el campo del número de secuencia contiene el ISN (Initial Sequence Number) elegido por el host para esa conexión. El número de secuencia del primer byte de datos será el ISN+1, ya que el bit SYN consume un número de secuencia.
- Número de acuse de recibo: Si el bit de control ACK está activo, este campo contiene los valores del siguiente número de secuencia que el receptor está esperando recibir.
- Longitud de cabecera: Tamaño de la cabecera.
- URG: Indica que el campo puntero urgente es significativo en este segmento.

- ACK: Indica que el campo de reconocimiento es significativo en este segmento. Todos los paquetes enviados después del paquete SYN inicial deben tener activo este bit.
- PSH: Función push. El receptor debe pasar los datos a la aplicación tan pronto como sea posible, no teniendo que esperar a recibir más datos.
- RST: Reinicio de la conexión.
- SYN: Sincroniza los números de secuencia.
- FIN: Ocupado para que el emisor solicite el cierre de la conexión.
- Tamaño de la ventana: Especifica el número máximo de bytes que pueden ser emitidos en el buffer de recepción.
- Checksum: Utilizado para la comprobación de errores tanto en la cabecera como en los datos
- Puntero urgente: Cantidad de bytes desde el número de secuencia que indica el lugar donde acaban los datos urgentes.
- Opciones: Se utiliza sólo en caso de necesitar añadir características no cubiertas por la cabecera fija.

1.4.4 Protocolo ICMP

Protocolo de Mensajes de Control y Error de Internet, ICMP, tiene un formato bastante simple y su utilidad es controlar si un paquete no puede alcanzar su destino, si su vida ha expirado, si es un paquete de eco o respuesta, etc. Se usa para manejar mensajes de error y de control necesarios para los sistemas de la red, informando con ellos a la fuente original para que evite o corrija el problema detectado. El protocolo ICMP sólo informa de incidencias en la entrega de paquetes o de errores en la red en general, pero no toma decisión alguna al respecto. La estructura genérica de un paquete ICMP se puede ver en la Figura 1-9.

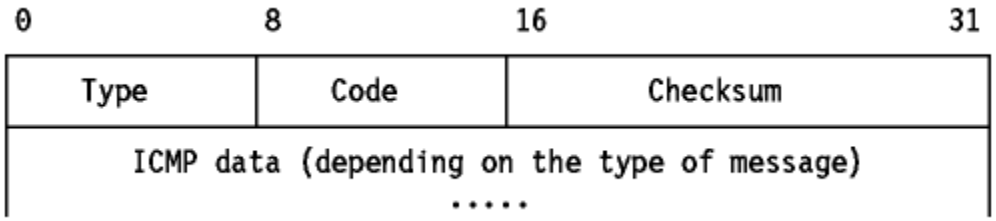

Figura 1-9 Paquete ICMP (fuente: http://mixteco.utm.mx).

El campo “Type” entrega mensajes informativos, de entre los cuales se pueden destacar: Si el campo “tipo” toma el valor 8, entonces se trata de un paquete de petición ECO, por el contrario, si tiene el valor 0, se trata de una respuesta de ECO. Las peticiones y respuestas de eco se usan en redes para comprobar si existe una comunicación entre dos hosts a nivel de capa de red, por lo que convierte a este protocolo en una gran herramienta para identificar fallos en este nivel, ya que verifica si la capa física (cableado), de enlace de datos y red, se encuentren en buen estado y configuración.

La Tabla 1-2 muestra los mensajes informativos del protocolo.

Tabla 1-2 Campo "Tipo" de un paquete ICMP

Valor	Mensaje
0	Echo Reply (respuesta de eco)
3	Destino inaccesible
4	Disminución del tráfico desde el origen
5	Redireccionar
8	Echo (solicitud de eco)
11	Tiempo excedido para un datagrama
12	Problema de parámetros
13	Solicitud de marca de tiempo
14	Respuesta de marca de tiempo
17	Solicitud de máscara de dirección
18	Respuesta de máscara de dirección

El campo “Code”, muestra posibles mensajes de error en la trama. Por ejemplo, en caso de obtener un mensaje ICMP de destino inalcanzable, con campo “Type” de valor 3, el error que se ha producido vendrá especificado en el campo “Code”. Los códigos de error con su respectiva explicación se encuentran en la Tabla 1-3.

Tabla 1-3 Campo "Code" de un paquete ICMP

Valor	Mensaje
0	No se puede llegar a la red
1	No se puede llegar al host o aplicación de destino
2	Destino no dispone del protocolo solicitado
3	No se puede llegar al puerto destino o la aplicación destino no está libre
4	Se necesita aplicar fragmentación, pero el flag correspondiente no lo indica
5	La ruta de origen no es correcta
6	No se conoce la red destino
7	No se conoce el host destino
8	El host origen está aislado
9	La comunicación con la red destino está prohibida
10	La comunicación con el host destino está prohibida
11	No se puede llegar a la red destino debido al tipo de servicio
12	No se puede llegar al host destino debido al tipo de servicio

1.4.5 Protocolo ARP

El protocolo ARP o protocolo de resolución de direcciones, es responsable de convertir la dirección de protocolo de alto nivel (direcciones IP) a direcciones de red físicas. Para que las direcciones físicas se puedan conectar con las direcciones lógicas, el protocolo ARP interroga a

los equipos de la red para averiguar sus direcciones físicas y luego crea una tabla de búsqueda entre las direcciones lógicas y físicas en una memoria caché [9].

La Figura 1-10 muestra una trama genérica ARP.

- Hardware address: Especifica el tipo de hardware. Ejemplo: Ethernet.
- Protocol address: Tipo de protocolo
- Hardware address length: Longitud en bytes de la dirección hardware del paquete
- Protocol address length: Longitud en bytes de las direcciones del protocolo en el paquete.
- Operation code: Especifica si se trata de una petición (1) o una solicitud (2)
- Source/Target hardware address: Contiene direcciones físicas del hardware
- Source/Target protocol address: Contiene las direcciones del protocol (Dirección IP)

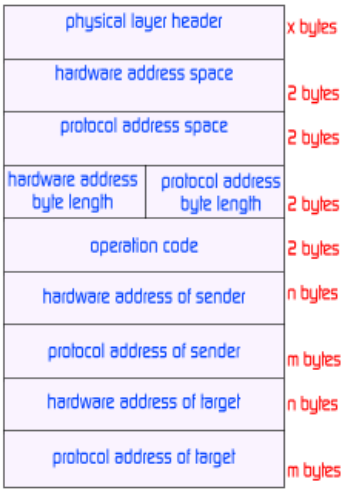

Figura 1-10 Paquete ARP (fuente: neo.lcc.uma.es).

1.4.6 Protocolo DNS

Sistema de nombres de dominio (Domain Name System), está ubicado en la capa de aplicación del modelo OSI y puede ser implementado tanto en TCP como UDP. Este protocolo se utiliza para recordar de manera sencilla las direcciones IP. Mediante este protocolo se asigna un nombre a una IP. Se puede decir que el protocolo DNS es un sistema jerárquico y distribuido que permite traducir nombres de dominio en direcciones IP y viceversa.

La Figura 1-11 muestra una trama DNS genérica.

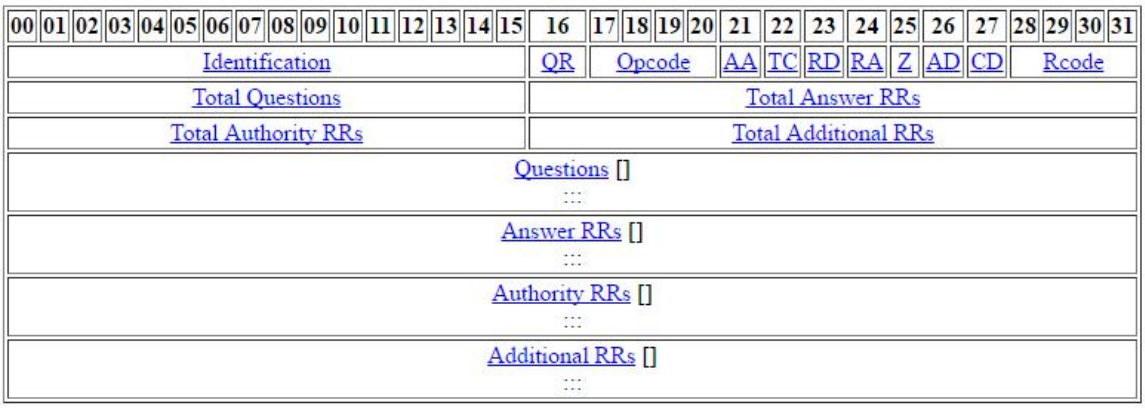

Figura 1-11 Trama DNS (fuente: neo.lcc.uma.es).

Algunos campos importantes:

- Identification: Campo usado para hacer coincidir los paquetes de petición, con los de respuesta
- QR: un “0” indica que es un paquete de petición, un “1” indica que es un paquete de respuesta
- Opcode: Código que indica la función del paquete. Ejemplo: “0” indica que es una petición estándar, “2” indica que es una petición de estado del servidor.
- Rcode: Código de retorno. Dependiendo del valor de este campo, puede indicar distintos errores en el paquete.

1.5 Análisis de sniffers existentes

1.5.1 Acrylic

Acrylic, es un completo software de pago mediante el cual se puede analizar el tráfico en una red de computadoras, además de tener varias aplicaciones extras para la optimización de una infraestructura Ethernet [3].

1.5.2 “Sniffing” con Acrylic

Una vez instalado el software, basta con seleccionar la tarjeta de red operativa en la máquina, seleccionar la opción de visibilidad de los paquetes que transitan la red, seleccionar el modo monitor (promiscuo) y apretar el botón de inicio como muestra la Figura 1-12.

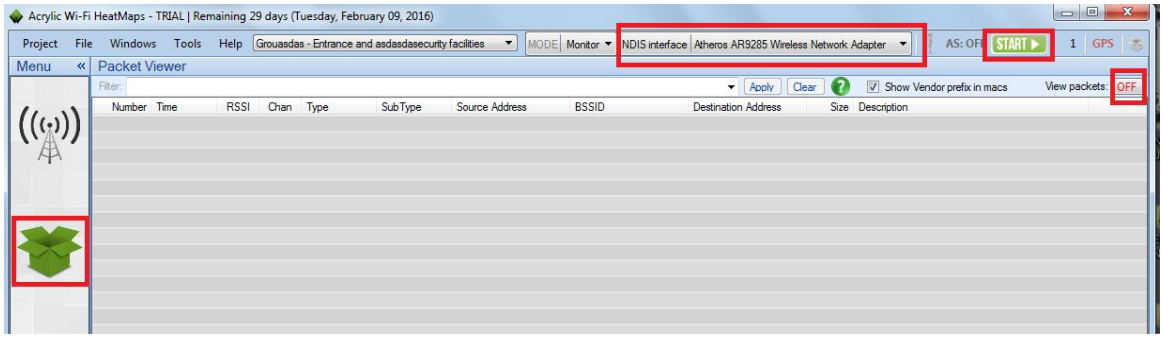

Figura 1-12 Sniffer Acrylic (fuente: propia).

Acrylic comenzará a recibir datos como se muestra en la Figura 1-13. Aquí se aprecian los distintos tipos de tramas que circulan en la red.

Number	Time	RSSI	Chan	Type	SubType	Source Address	BSSID	Destination Address	Size	Description
7004	182.9444	-38	7	Control	Acknowledges				35	
7005	182.9458	-39	7	Data	Data	Shenzhen_78:20:D2	Shenzhen_78:20:D3	ASRockIn_37:07:8F	123	
7006	182.9471	-39	7	Data	Data	Shenzhen_78:20:D2	Shenzhen_78:20:D3	ASRockIn_37:07:8F	123	
7007	182.9724	-77	7	Data	Data	ASRockIn_37:07:8F	Shenzhen_78:20:D3	Shenzhen_78:20:D2	19	
7008	182.9727	-39	7	Control	Acknowledges				35	
7009	182.9737	-38	7	Data	Data	Shenzhen_78:20:D2	Shenzhen_78:20:D3	ASRockIn_37:07:8F	123	
7010	182.9863	-78	7	Data	Data	ASRockIn_37:07:8F	Shenzhen_78:20:D3	Shenzhen_78:20:D2	19	
7011	182.9866	-38	7	Control	Acknowledges				35	
7012	182.9879	-39	7	Data	Data	Shenzhen_78:20:D2	Shenzhen_78:20:D3	ASRockIn_37:07:8F	123	
7013	183.0022	-39	7	Data	Data	Shenzhen_78:20:D2	Shenzhen_78:20:D3	ASRockIn_37:07:8F	123	
7014	183.0040	-40	7	Data	Data	Shenzhen_78:20:D2	Shenzhen_78:20:D3	ASRockIn_37:07:8F	123	
7015	183.0057	-38	7	Data	Data	Shenzhen_78:20:D2	Shenzhen_78:20:D3	ASRockIn_37:07:8F	123	
7016	183.0071	-39	7	Data	Data	Shenzhen_78:20:D2	Shenzhen_78:20:D3	ASRockIn_37:07:8F	123	
7018	183.0107	-57	7	Control	Acknowledges				35	
7017	183.0107	-40	7	Data	QoSData	Shenzhen_78:20:D2	Shenzhen_78:20:D3	D-LinkCo_55:95:D3	129	
7019	183.0111	-59	7	Data	QoSData	D-LinkCo_55:95:D3	Shenzhen_78:20:D3	Shenzhen_78:20:D2	117	
7020	183.0112	-43	7	Control	Acknowledges				35	
7021	183.0153	-58	7	Data	QoSData	D-LinkCo_55:95:D3	Shenzhen_78:20:D3	Shenzhen_78:20:D2	21	
7022	183.0153	-58	7	Control	Acknowledges				35	

Figura 1-13 Obtención de tramas con Acrylic (fuente: propia).

Haciendo click en una potencial trama a estudiar, Acrylic muestra el “árbol” del paquete, que incluye la estructura de este, indicando el orden y el significado de los bytes que componen la trama. Además incluye una vista Hexadecimal del paquete, como se muestra en la Figura 1-14.

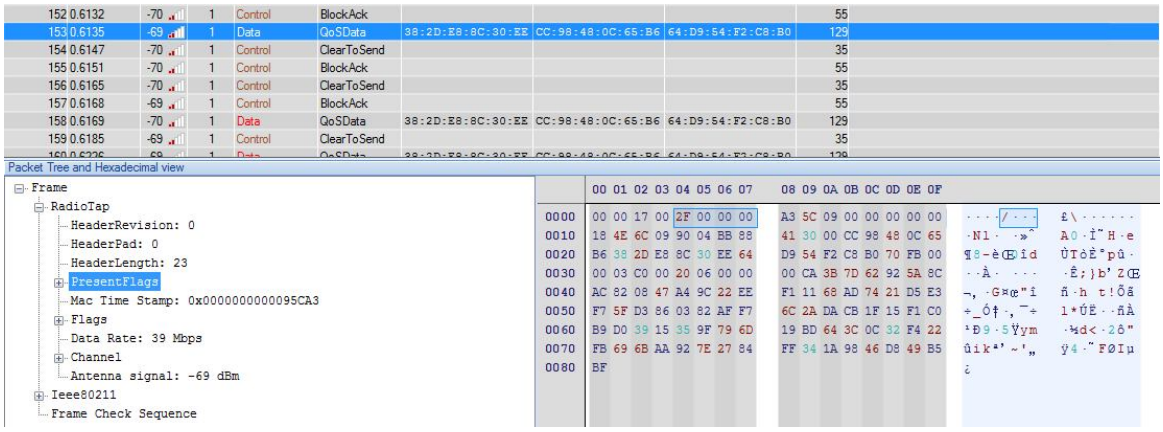

Figura 1-14 Árbol del paquete y vista hexadecimal (fuente: propia).

1.5.3 Análisis de los datos obtenidos con Acrylic

Mediante esta herramienta, se puede obtener la estructura principal de las distintas tramas en una red, como la cantidad de bytes que componen una trama y la posición que ocupa cada byte en esta.

La Tabla 1-4 muestra algunos de los resultados obtenidos.

Tabla 1-4 Datos obtenidos de Acrylic

Tipo de trama	Sub tipo	Tamaño (bytes)	Bytes 0-3	Byte 17	Bytes 18-19		Bytes 20-21
Control	RTS	43	Cabecera	Data rate	Frec. canal	Del	BW canal
Control	CTS	35	Cabecera	Data rate	Frec. canal	Del	BW canal
Control	ACK	35	Cabecera	Data rate	Frec. canal	Del	BW canal
Data	QoSData	Variable	Cabecera	Data rate	Frec. canal	Del	BW canal
Management	Beacon	Variable	Cabecera	Data rate	Frec. canal	Del	BW canal

1.6 Wireshark

1.6.1 Introducción al software

Wireshark es otro software para el análisis de tráfico de redes, del tipo gratuito.

La ventaja de este software, es la mejor indicación del lugar que ocupan los bytes en la trama y la posibilidad de ver los bytes tanto en forma binaria como hexadecimal. En la Figura 1-15, se observa una trama TCP en Wireshark en código binario.

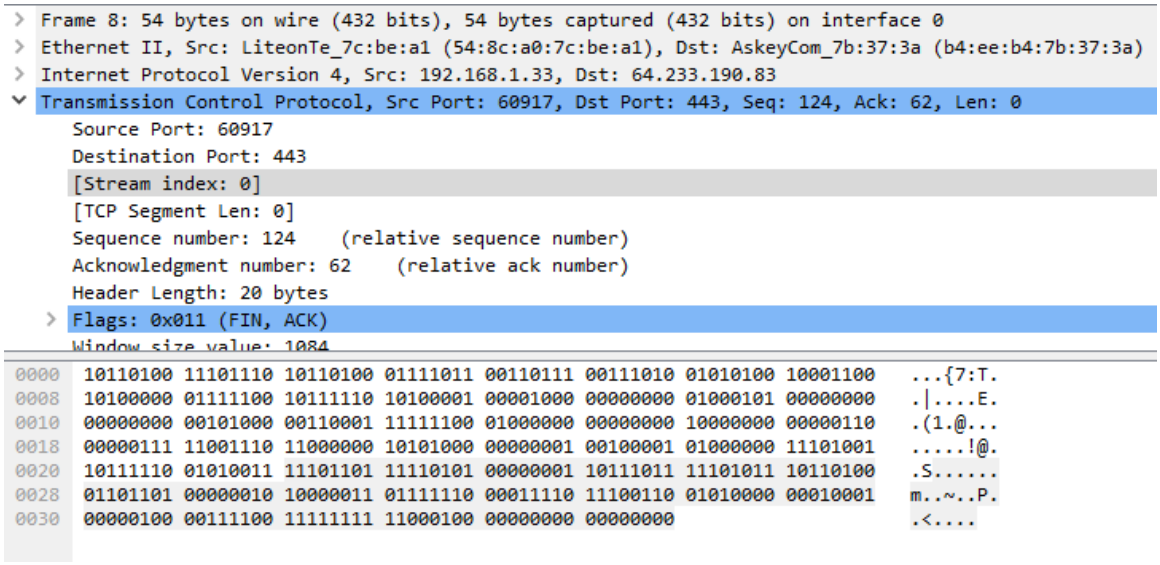

Figura 1-15 Trama TCP en Wireshark (fuente: propia).

1.7 Análisis de Acrylic y Wireshark

En la Tabla 1-5, se encuentra una breve reseña de los programas.

Tanto Wireshark como Acrylic y cualquier analizador de paquetes en general, tienen la desventaja de no poder hacer el “sniffing” y tener conexión Wi-Fi al mismo tiempo, dado que se fuerza a la tarjeta a operar a una frecuencia distinta a la del funcionamiento de la red.

Mediante la Tabla 1-5, es necesario rescatar las ventajas de estos softwares para ser consideradas en el desarrodo del sniffer. Se deberá contar con una interfaz gráfica sencilla, que permita la discriminación de tramas irrelevantes.

Tabla 1-5 Comparativa de analizadores de paquetes

	Acrylic	Wireshark
Ventajas	Ofrece interfaz gráfica sencilla. Dotado de filtros capaces de discriminar tramas innecesarias. Rápida obtención de datos (Orden de los milisegundos).	Ofrece interfaz gráfica sencilla. Dotado de filtros capaces de discriminar tramas innecesarias. Rápida obtención de datos (Orden de los milisegundos). Mayores prestaciones a la hora de analizar datos.
Desventajas	No es posible analizar tráfico y mantener conexión Wi-Fi simultáneamente.	No es posible analizar tráfico y mantener conexión Wi-Fi simultáneamente.

1.8 Problemática y abordaje de la solución

Si bien ya existen analizadores de tramas en el mercado, no se cuenta con alguno que sea indetectable. Por lo general, y debido a la gran cantidad de procesos de los distintos sistemas operativos existentes, es normal que un ordenador esté en constante transmisión y recepción de tramas, lo que automáticamente vuelve detectable al equipo en el cual se ocupe el sniffer. Este estado de invisibilidad, puede ser de gran ayuda al momento de detectar ataques maliciosos a una red, o bien consultar el tráfico de los usuarios pertenecientes a la red sin ser detectado.

Se necesita de un sistema operativo sencillo, libre de procesos innecesarios que permita la captura de tramas sin ser detectado, además, debe contar con un compilador que tenga la capacidad de acceder a los registros de la tarjeta de red mediante un bus PCI, esto con el fin de poder configurar la tarjeta en su modo “promiscuo”. Por otra parte, la tarjeta de red, debe ser de tipo PCI y tener la documentación suficiente, para así poder configurar su funcionamiento. La idea final, es que la tarjeta de red entre en el modo promiscuo y comience a “escuchar” todo el tráfico de la red, esto debe ser logrado únicamente recibiendo las tramas y no enviando dato alguno de vuelta.

1.9 Objetivos

El objetivo general del proyecto es lograr el desarrollo de un sniffer indetectable dentro de una red, el cual posea una interfaz sencilla y amigable para el usuario.

Los objetivos específicos, comprenden el reconocimiento, por parte del sniffer, de la mayor cantidad de protocolos posibles. Además del reconocimiento de protocolos, se pretende lograr un análisis detallado de cada elemento que se encuentre presente dentro de las tramas. Así se contará con una herramienta de análisis completa.

1.10 Solución propuesta

En respuesta a las necesidades anteriormente planteadas, se opta por trabajar en el sistema operativo DANUX, diseñado por el profesor David Velasco López. DANUX permite acceder a los registros de una tarjeta de red, siempre y cuando ésta sea de bus PCI, además, puede ser configurada de tal forma, que sólo esté recibiendo tramas, sin hacer envío alguno, lo cual es la condición principal para obtener un sniffer indetectable. La tarjeta de red será una Realtek RTL8169, debido a su conector PCI y a que es uno de los chips mayormente documentados.

2 Estructura del proyecto

2.1 Controlador RTL8169

El RTL8169 es un controlador integrado de acceso al medio Gigabit Ethernet de tipo PCI, para usar en adaptadores de red en servidores o computadores personales [10]. Implementa un bus de interface de 33/66 MHz, 32/64-bit PCI v2.2 para comunicación entre hosts y cumple con las especificaciones de la IEEE 802.3 para 10/100 Mbps de conexión ethernet y la especificación de la IEE 802.3z para 1000 Mbps.

Algunas de sus características principales son [10]:

- Soporta auto negociación para 10 Mbps, 100 Mbps y 1000 Mbps.
- Controlador ethernet con bus de tipo PCI
- Soporta modos “big endian” y “little endian”
- Soporta bus de tipo 32 y 64 bits
- Función “Wake-On-LAN” y “remote wake-up”
- Incluye la posibilidad de programación tanto de la ráfaga DMA, como del umbral de transmisión y recepción
- “Timer” para propósitos generales de 32 bits para generar interrupciones
- Capacidad de operación en half dúplex y full dúplex.

La Figura 2-1 muestra una imagen de una tarjeta de red con el chip RTL8169 incorporado.

2.1.1 Registros RTL8169

Para la programación adecuada de la tarjeta de red, es necesario el conocimiento de sus registros más importantes, los cuales se describen en la Tabla 2-1.

Figura 2-1 Tarjeta de red con chip RTL8169 (fuente: spelesprieks.lv).

Tabla 2-1 Registros principales RTL8169

Registro	Descripción
IDR0 – IDR5	Los registros IDR0 hasta IDR5 guardan la dirección MAC de la tarjeta de red.
TCR	Registro de configuración de la transmisión de datos.
RCR	Registro de configuración de la recepción de datos.
9346CR	Este registro permite seleccionar el modo de operación del chip RTL8169 (Modo normal, auto cargar configuración por defecto, modo de programación y activar la escritura en los registros de configuración del chip.
CONFIG0 – CONFIG 5	Registros de configuraciones varias del chip RTL8169, para acceder a ellos, es necesario activar la escritura en el registro 9346CR.
RDSAR	Dirección de inicio del descriptor de recibimiento de paquetes.
RMS	Define el tamaño máximo de paquetes a recibir.

2.2 Entorno de trabajo

El entorno a trabajar es una red LAN con topología de bus. Se ocupa un hub como método de interconexión de dispositivos, dada su característica simplificada con respecto al switch, el hub envía las tramas a todos los dispositivos conectados a la red, en comparación al switch, que envía los datos provenientes de la computadora de origen solamente a la computadora de destino. Esto se debe a que los switches crean una especie de canal de comunicación exclusiva entre el origen y el destino. Esta característica del hub, ayuda al sniffer en la captura de tramas.

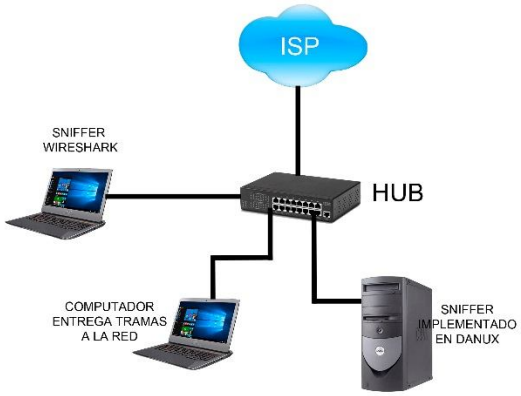

Figura 2-2 Estructura del proyecto (fuente: propia).

Como se muestra en la Figura 2-2, se conectan dos computadores portátiles al hub, uno será el encargado de enviar datos a la red, los que deben ser capturados por el sniffer, el segundo tendrá instalado el software Wireshark, el cual es un sniffer usualmente utilizado, y que cumplirá la función de comparar los datos que capture, con los datos obtenidos por el sniffer implementado en DANUX.

2.3 Estructura del analizador de paquetes en DANUX

El sniffer desarrollado en DANUX, consta de los siguientes archivos:

1. SNIFF_50: Archivo principal, contiene los procedimientos de inicialización de la tarjeta de red RTL8169, llenado de descriptores de la misma, reconocimiento de protocolos, escritura en la memoria de la pantalla.
2. RTL_8169: Adjunta las funciones de cada registro perteneciente a la tarjeta de red RTL8169 como constantes.
3. TCP_VAR: Incluye la dirección especifica de los puertos de los registros pertenecientes a la tarjeta de red, además de variables utilizadas en el archivo principal SNIFF_16.
4. TCP_TYP: Define variables ubicadas dentro de las tramas capturadas, esto permite hacer el llenado de los campos de cada trama, en la pantalla del sniffer.

El contenido de los archivos RTL_8169, TCP_VAR y TCP_TYP, son incluidos mediante la función “include” en el archivo principal SNIFF_50.

2.4 Inicialización de la tarjeta de red

La inicialización de la tarjeta se lleva a cabo en el archivo principal y se muestra en el Listado 2-1.

Listado 1 Inicialización de la tarjeta de red

```
1  Begin  { Init_NIC  }
2      For NIC_Cnt := 1 to  max_NIC DO  ;
3         Begin
4
5          Command := RTL_NIC_Reset;
6          While Command = RTL_NIC_Reset Do;
7
8          Cfg9346  := Cfg9346_Unlock;
9          Config1  := Config1 Or Power_Manager;
10         Config2  := PCI_Bus_32_Bits;
11         Config5  := Enable_Reset_PCI;
12         Cfg9346  := Cfg9346_Lock;
13
14
15         IntrStatus := $FFFF; {reseteo de todos los bits}
```

Para inicializar la tarjeta, primero se requiere cambiar el modo de operación de ésta, ajustándola al modo de habilitación de escritura en los registros de configuración. Para este se debe acceder al registro 9346CR y establecer un 1 lógico en los bits 7 y 6, como se muestra en la Tabla 2-2.

Tabla 2-2 Registro 9346CR

Bit	R/W	Símbolo	Descripción		
7:6	R/W	EEM1-0	EEM1	EEM0	Modo de operación
			0	0	Normal
			0	1	Auto-load: Dura 2 ms, una vez completado, el chip vuelve al modo normal y los valores de todos los demás registros vuelven a sus valores por defecto.
			1	0	Programming: Las operaciones en el bus, tanto de la red como del host se encuentran deshabilitadas.
			1	1	Config Register Write Enable: Para acceder a la escritura en los registros CONFIG0 hasta CONFIG5, se debe acceder a este modo.

Luego, se accede al bit 0 del registro CONFIG1 para habilitar la administración de poder. Se debe establecer un 1 lógico en este bit. En el registro CONFIG2, se debe establecer que el ancho del bus PCI es de 32 bits, estableciendo un 0 lógico en el bit 3 de dicho registro. Posteriormente, se establece un 1 lógico en el bit PME_STS del registro CONFIG5, el cual permite reiniciar el chip mediante software. Una vez hecho esto, se vuelve a colocar el chip en el modo de operación normal, estableciendo en 0 los bits 6 y 7 del registro 9346CR. Una vez terminado el proceso, se reinician todos los bits del registro ISR (Interrupt status register).

2.5 Asignación de direcciones DMA

Se debe indicar el buffer del descriptor de recepción de tramas al chip RTL8169. Se asigna una dirección de memoria al registro RDSAR (Receive Descriptor Start Address Register), luego se procede al llenado de descriptores como se muestra en el Listado 2-2.

Listado 2 Llenado de descriptores

```
1         RxDescAddrHigh := $00000000;
2         RxDescAddrLow  := @descrip_RX;
3
4   Procedure Init_Buffer_NIC;
5    Begin
6       {*********** Llenado de descriptores RX  ***********}
7       For i := 1 to RX_Max  do
8           Begin
9           descrip_RX [i].status     := OWNbit Or MAX_SIZE;
10          descrip_RX [i].buff_haddr := $00;
11          descrip_RX [i].vlan_tag   := $0;
12          descrip_RX [i].buff_addr  := @trama_in [i];
13          End;
       descrip_RX [RX_Max].status := descrip_RX [RX_Max].status Or
                                        EORbit;
    End;
```

2.6 Habilitación de recepción de tramas

El bit que habilita la recepción de tramas, se ubica en el registro Command, el cual debe ser ajustado a un uno lógico. La Tabla 2-3 muestra el registro y su contenido.

Tabla 2-3 Registro Command

Bit	R/W	Símbolo	Descripción
3	R/W	RE	Receiver enable: Activa la función de recibir datos estableciendo un uno en este bit.
2	R/W	TE	Transmiter enable: Al poner este bit en uno, se activa la función de transmitir datos.

Luego, se establece la configuración para la recepción de tramas, mediante el registro RCR, el cual se encuentra detallado en la Tabla 2-4.

Tabla 2-4 Registro RCR

Bit	R/W	Símbolo	Descripción
31-25	-	-	Reservados
24	R/W	MulERINT	Cuando este bit es puesto en uno, todo paquete recibido invoca a una interrupción.
23-16	-	-	Reservados
15-13	R/W	RXFTH2,1,0	Rx FIFO Threshold: Especifica el nivel de umbral de la recepción de la cola FIFO (First in, first out). Cuando el número de bytes recibidos de un paquete que está siendo recibido en la cola FIFO del RTL8169, ha alcanzado este nivel, los datos serán transferidos desde la cola FIFO, hasta la memoria del host. Este campo ajusta el nivel del umbral, acorde a la siguiente tabla: 010: 64 bytes 011: 128 bytes 100: 256 bytes 101: 512 bytes 110: 1024 bytes 111: Sin umbral de Rx
12-11	-	-	Reservados
10-8	R/W	MXDMA2,1,0	Max DMA Burst size: Este campo ajusta el tamaño máximo de la ráfaga DMA (Direct Memory Access).
7	-	-	Reservado
6	R	9356SEL	Identifica el tipo de EEPROM usado.
5	R/W	AER	Accept Error Packet: 1: Acepta todos los paquetes con error. 0: Rechaza todos los paquetes con error.
4	R/W	AR	Accept Runt: Permite recibir paquetes más pequeños que 64 bytes. El paquete deber ser al menos de 8 bytes.
3	R/W	AB	Acepta o rechaza paquetes broadcast.
2	R/W	AM	Acepta o rechaza paquetes multicast.
1	R/W	APM	Acepta o rechaza paquetes con similitudes físicas.
0	R/W	AAP	Acepta o rechaza paquetes provenientes de todas las direcciones (Modo promiscuo).

En el registro de configuración de recepción, se pueden filtrar los tipos de tramas que se pueden recibir, ya sean de control, datos o de administración, definir si se aceptan o no los paquetes broadcast, multicast y unicast, filtrar paquetes con errores, definir la velocidad del DMA Burst, o bien activar el modo promiscuo mediante la función AAP (Accept Destination Address Packets).

La programación en el archivo principal queda como se muestra en el Listado 2-3.

Listado 2-3 Configuración de recepción y llenado de descriptores

```
        {*************************** DMA SET ****************************}
        {*  RDSAR: Receive Descriptor Start Address, OFFSET E4-EB           *}
        {*  offset E7-E4: low 32 bit addr     Asignando direcciones DMA.    *}
        {*  offset E8-EB: high 32 bit addr                                  *}
        {*  Indicar buffers de descriptores a la tarjeta                    *}
        {****************************************************************}
        RxDescAddrHigh := $00000000;
        RxDescAddrLow  := @descrip_RX;
        TXdescAddrHigh := $00000000;
        TXdescAddrLow  := @descrip_TX;

        Writeln('****************** RX TX ENABLE! ****************');
        Command := Cmd_Rx_Enb Or Cmd_Tx_Enb;
        Early_Tx_Threshold := Threshlod_Level;



        Rx_MaxSize := max_size;

        Rx_Config:=  no_Theshold           Or
                      RX_DMA_Burstno_limit Or
                        Accept_Broadcast   Or
                         Accept_My_Phys    Or
                          Accept_multicast Or
                            Promiscuous  ;


Procedure Init_Buffer_NIC;
 Begin
    {*********** Llenado de descriptores RX  ***********}
    For i := 1 to RX_Max  do
        Begin
        descrip_RX [i].status     := OWNbit Or MAX_SIZE;
        descrip_RX [i].buff_haddr := $00;
        descrip_RX [i].vlan_tag   := $0;
        descrip_RX [i].buff_addr  := @trama_in [i];
        End;
    descrip_RX [RX_Max].status := descrip_RX [RX_Max].status Or
                                     EORbit;
End;
```

3 Desarrollo del Sniffer

3.1 Producción principal

En el Listado 3-1, se aprecia el inicio de la producción principal, se asignan valores a algunas variables, la variable “RUN” en la línea 20 se torna valida y se inicia “pantalla_alt” (línea 22), la cual muestra en pantalla las tramas en su totalidad.

Listado 3-1 Inicio de la producción principal

```
1   Desde
2
3      limpia_pantalla ;
4      writeln (my_mac);
5
6      { Iniciacion de variables y de la tarjeta *****}
7      max_NIC := 1;
8      {  writeln ( 'Cantidad de dispositivos Realtek 8169: ' , max_NIC );}
9
10      Init_Buffer_NIC; {  descriptores RX TXN TXH y ubicacion de buffers }
11      Init_Nics;
12      Init_Nics;
13      {test;}
14
15      i     := 1;
16      i_tx  := 0;
17      trama_count  := 0 ; { tramas capturadas }
18      trama_actual := 1 ;
19
20      RUN     := True;
21      status := 'R U N   ';
22      pantalla_alt := True ;
```

Listado 3-2 Botones del procedimiento principal

```
1   repeat
2         Desde
3               Desde
4                    ? (scan = page_up) And
5                      (trama_count >trama_actual) And (Not Run);
6                      trama_actual := trama_actual +1;
7                      dump_trama;
8               Sino;
9                     ? (scan = page_down) And (trama_actual >1) And (Not Run);
10                     trama_actual := trama_actual -1;
11                     dump_trama;
12              Sino
13                   ? (scan = inicio ) And (Not Run);
14                     trama_actual := 1 ;
15                     dump_trama;
16              Sino
17                   ? (scan = fin ) And (Not Run);
18                     trama_actual := trama_count ;
19                     dump_trama;
20              Sino
21                   ? (scan = Backspace) ;
22                     trama_actual  := 1 ;
23                     trama_count   := 0 ;
24                     dump_hexa (top_outH,trama_count_H,1);
25                     dump_hexa (top_outL,trama_count_L,1);
26                     limpia_pantalla ;
27              Sino
28                   ? (scan = pausa);
29                     run := NOT RUN ;
30                     IF Run
31                      Then status := 'R U N   '
32                      Else status := 'S@T@O@P@';
33
34                     IF Not Run
35                      Then
36                       Begin
37                        Stop_nic  ;
38                        dump_trama ;
39                        scan := null;
40                       End
41                      Else
42                       Begin
43                        init_nics ;
44                        init_nics ;
45                       End;
46              Sino
47                   ? (scan = F1) And (Not Run);
48                     pantalla_alt :=  Not pantalla_alt ;
49                     dump_trama ;
50              sino
51                   ? (scan =esc)
52
53              Sino { ojo es muy importante colocar esta opcioon vacia, }
54                     { sino declara que toda la opcion principal no      }
55                     { cumple y se sale del programa                     }
56                     scan:= null;
57            Hasta ;
```

En el Listado 3-2 se ven las distintas opciones que tiene el analizador de paquetes. Desde la línea 4 hasta la 7, se aprecia que mientras el programa no esté corriendo (“Not run”), al apretar “page up” o “page down”, es posible navegar a través las tramas capturadas. En la línea 47, al apretar F1, y mientras no se esté ejecutando la captura de tramas, se tiene un cambio de pantalla. La pantalla

principal se muestra en la Figura 3-1, la cual muestra la trama capturada en su totalidad, tanto en hexadecimal, como su versión ASCII a la derecha. Arriba, se aprecia “Trama tope”, que es el número de trama capturada, “STOP” dado que el programa se encuentra en estado “Not run””, “Size” muestra el tamaño de la trama actualmente consultada.

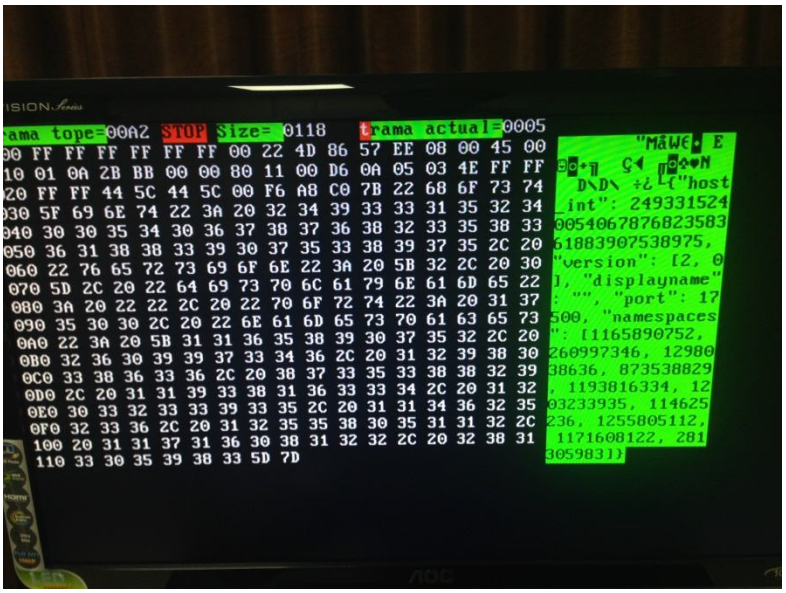

Figura 3-1 Pantalla captura de tramas (fuente: propia).

Al apretar “F1”, el programa entra en una segunda pantalla que es mostrada en la Figura 3-2. En este punto, se muestra la información más detallada del paquete, haciendo alusión a la MAC de fuente y destino, identificación del protocolo, puertos utilizados, IP de fuente y destino, largo del paquete, entre otros atributos.

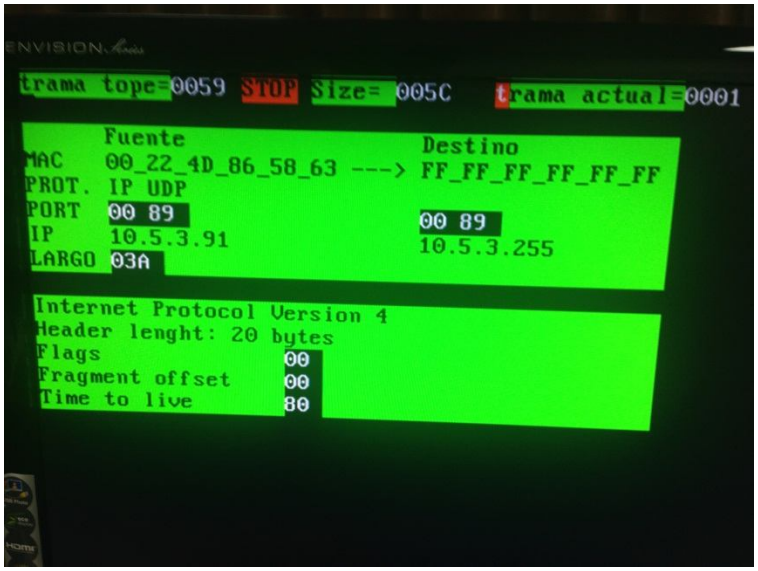

Figura 3-2 Pantalla secundaria (fuente: propia).

3.1.1 Reconocimiento de protocolos

Esta parte del proyecto se basa principalmente en la sección de reconocimiento de protocolos, correspondiente al archivo principal.

Listado 3-3 Identificación de protocolo

```
1   head_type := trama_in [trama_actual] | tipo;
2        Desde
3             ? head_type = protocolo_IP;
4               { ************************}
5               { analisis de protocolos IP }
6               { ************************}
7               Desde
8                    ? trama_in [trama_actual] | protocolo =protocolo_ICMP;
9                      pal_aux := 'PROT. IP ICMP  ';
10              Sino
11                   ? trama_in [trama_actual] | protocolo =protocolo_TCP;
12                     pal_aux := 'PROT. IP TCP  ';
13              Sino
14                   ? trama_in [trama_actual] | protocolo =protocolo_UDP;
15                     pal_aux := 'PROT. IP UDP ';
```

Como se puede ver en la línea 1 del Listado 3-3, “head_type” es una variable de tipo Word, la cual es igualada a “trama_in [trama_actual] | tipo”. “tipo” es una variable de tipo Word, ubicada en el byte 12 y 13 de una trama determinada, “trama_in [trama_actual]” sitúa la trama actualmente recibida. Por lo tanto, la variable “head_type” toma el valor del byte número 12 y 13, de la trama actual, el cual indica el tipo de protocolo ethernet. Luego, éste es comparado con la constante “protocolo_IP” (línea 3), la cual está definida como una constante ($0008) y es el valor a esperar si es que la trama recibida es del tipo IPv4. Si esta condición se cumple, se entra a la línea 7 para determinar el tipo de protocolo IP de manera similar.

3.2 Planteamiento

Se pretende analizar cada protocolo de manera separada, es decir, una vez reconocido el protocolo, crear el procedimiento que llene la pantalla secundaria, con los datos extraídos de las tramas recibidas como indica el Listado 3-4. Para este ejemplo se toma el protocolo UDP como referencia, se aprecia en la línea 1 hasta la 3, del Listado 3-4, el procedimiento para detectar el tipo de protocolo, una vez reconocido, de la línea 4 en adelante están las acciones que extraen los datos de la trama.

Listado 3-4 Extracción de datos trama UDP

```
1  Sino
2      ? trana_in [trama_actual] | protocolo = protocolo_UDP;
3        pal_aux := ‘ PROT. IP UDP ‘;
4        dump_palabra (VGA [5],pal_aux,47);
5        pal_aux := ‘ LARGO ‘ ;
6        dump_palabra (VGA[8], pal_aux,47);
7  dump_hexa (VGA [8][13], trama_in [trama_actual] | LARGO_UDP_H_1);
8  dump_hexa (VGA [8][15],trama_in [trama_actual] | LARGO_UDP_L,1);
9
10 VH := trama_in [trama_actual] | version_header;
11 If VH = $45 Then
12      Begin
13          pal_aux := ‘ Internet Protocol Version 4 ‘;
14          dump_palabra (VGA [10],pal_aux,47);
15          pal_aux := ‘ Header length: 20 bytes ‘;
16          dump_palabra (VGA [11],pal_aux,47);
17      pal_aux := ‘ FLAGS ‘;
18      dump_palabra (VGA [12],pal_aux,47);
19      dump_hexa (VGA [12][39],trama_in [trama_actual] | ipv4_flags,1);
20      pal_aux := ‘ Fragment offset ‘;
21      dump_palabra (VGA [13],pal_aux,47);
22      dump_hexa (VGA [13][39],trama_in [trama_actual] | ipv4_offset,1);
23      pal_aux := ‘ Time to live ‘;
24      dump_palabra (VGA [14],pal_aux,47);
25      dump_hexa (VGA [14][39],trama_in [trama_actual] | ipv4_time_live,1);
26      End;
```

3.3 Programación final del sniffer

Debido a la inestabilidad del sistema, fue requerida una nueva versión del sistema operativo DANUX. El sistema se volvía ineficiente al hacer tantas llamadas a la memoria de la pantalla VGA, así, si se continuaba con la programación del sniffer, tarde o temprano el programa colapsaría, lo cual podría haber resultado en pérdida de tramas en el proceso de “sniffing”. Esta nueva versión, permite aplicar las funciones necesarias sin tantas líneas de programación, para así dar un mayor rendimiento al sniffer.

3.3.1 Programa principal

Se explica de manera general cómo funciona el reconocimiento de tramas y cómo se muestran sus datos en pantalla.

3.3.2 Producciones

Se crean varias producciones, las cuales pueden ser llamadas en cualquier momento en el programa.

En el Listado 3-5 se aprecia la Producción “limpia_pantalla”, la cual borra todo lo que se encuentra en pantalla mediante la línea 3 y vuelve a llenar con los parámetros comunes para todos los protocolos (MAC de fuente y destino, IP de fuente y destino, contador de tramas) a partir de la línea 5. Esta producción permite que, cada vez que consultemos una trama, se borren los parámetros escritos en pantalla de la trama anterior, y se escriban los de la trama actual.

En la Figura 3-3 se puede ver el resultado de la producción “limpia_pantalla” en el monitor.

Figura 3-3 Producción limpia_pantalla (fuente: propia).

Listado 3-5 Producción limpia_pantalla

```
1  Produccion limpia_pantalla ;
2   Desde
3    Fill (VGA[3],00,4000) ;
4
5    VGA [2][1]   := Char ($07)+ 'Trama Tope= $' ;
6    Dump_Word (top_out,trama_count);
7
8    VGA [2][2]   := Char ($07)+ 'Trama Actual= $' ;
9    Dump_Word (act_out,trama_actual);
10
11   VGA [3][1] := ' Source MAC ';
12   macF       :=  trama_in [trama_actual] | SOU_mac ;
13   VGA [3][2] :=  char ($07)+#MacF;
14
15   VGA [3][3] := ' Destination MAC';
16   macD     :=  trama_in [trama_actual] | DES_Mac ;
17   VGA [3][4] :=  char ($07)+#MacD;
18
19   VGA [4][1] := ' Source IP';
20   ipF      := trama_in [trama_actual] | SOU_IP ;
21   VGA [4][2] :=   char ($07)+#ipF;
22
23   VGA [4][3] := ' Destination IP';
24   ipD      := trama_in [trama_actual] | DES_IP ;
25   VGA [4][4] :=   char ($07)+#IpD;
26
27 Hasta;
```

En el Listado 3-6 se muestran varias producciones, las cuales tienen la finalidad de escribir en pantalla los campos a llenar para cada protocolo, esto se hace mediante el comando <nombre_produccion>;

Listado 3-6 Producciones

```
1   Produccion trama_ICMP;
2    Desde
3      VGA [14][1] := ' Type';
4      VGA [15][1] := ' Code';
5      VGA [16][1] := ' Checksum';
6      VGA [17][1] := ' Identifier(BE)';
7      VGA [18][1] := ' Identifier(LE)';
8      VGA [19][1] := ' Sequence number(BE)';
9      VGA [20][1] := ' Sequence number(LE)';
10   Hasta;
11
12  Produccion trama_ARP;
13   Desde
14     VGA [7][1]  := ' Hardware type';
15     VGA [8][1]  := ' Protocol type';
16     VGA [9][1]  := ' Hardware size';
17     VGA [10][1] := ' Protocol size';
18     VGA [11][1] := ' Opcode';
19   Hasta;
20
21  Produccion trama_DNS;
22   Desde
23     VGA [6][3] := ' Transaction ID';
24     VGA [7][3] := ' Flags';
25     VGA [8][3] := ' Question';
26     VGA [9][3] := ' Answer RRS';
27     VGA [10][3] := ' Authority RRS';
28     VGA [11][3] := ' Additional RRS';
29   Hasta;
30
31  Produccion trama_TCP;
32   Desde
33     VGA [14][1] := ' Source Port';
34     VGA [15][1] := ' Destination Port';
35     VGA [16][1] := ' Sequence Number';
36     VGA [17][1] := ' Acknowledgment Number';
37     VGA [18][1] := ' Flags';
38     VGA [19][1] := ' Window Size';
39     VGA [20][1] := ' Checksum';
40     VGA [21][1] := ' Urgent Pointer';
41   Hasta;
```

3.4 Reconocimiento trama IPv4

En la línea 262 hasta la 294 del Listado A-1, se muestra el reconocimiento de la trama IPv4. “head_type” se define como una variable de tipo Word en el archivo “TCP__VAR” y es igualada al campo “tipo” en la línea 263 del apéndice A-1, “tipo” se define como una variable de tipo Word, ubicada en el octeto número 12 de la trama actual, este octeto es el identificador del tipo de trama. Se tiene que “protocolo_IP” es una constante con valor “h0800” (hexadecimal), dicho valor es el indicador de que la trama es de tipo IPv4. Entonces, si “head_type” es igual a “protocolo_IP”, estamos en frente de una trama IPv4 y se procede a entrar en el “case”. Las líneas 268 a 275 llenan la pantalla, en la primera columna, con el nombre de cada campo presente en el header de una trama IPv4. A partir de la línea 277, hasta la 294, se procede a llenar en la columna 2, con sus respectivos valores numéricos, las variables del header Ipv4.

La Figura 3-4 muestra el resultado en pantalla. Notar que los valores mostrados están en formato decimal.

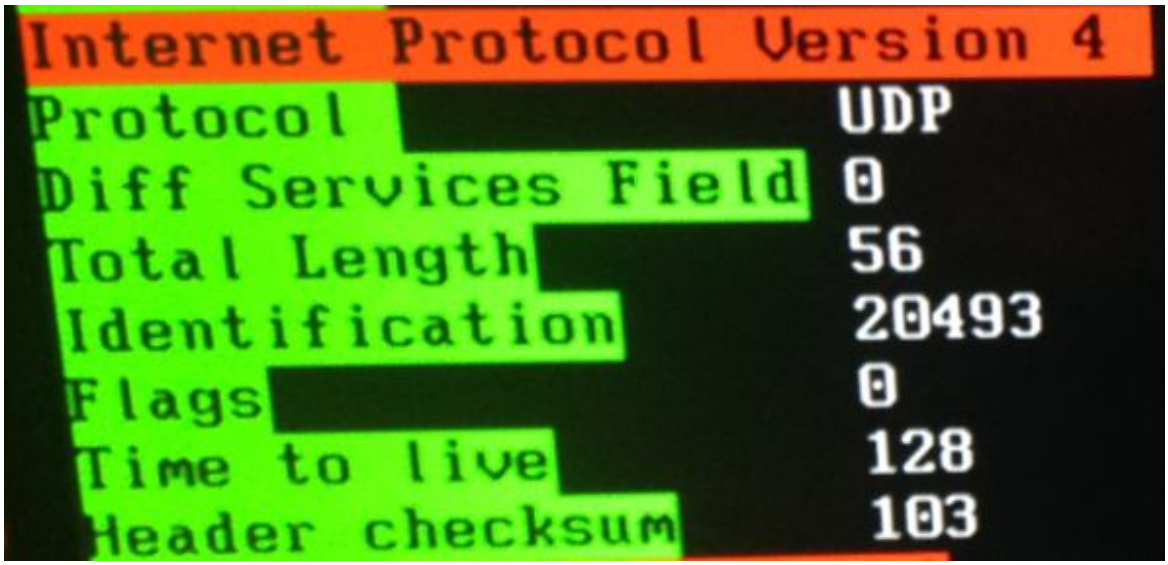

Figura 3-4 Trama IPv4 (fuente: propia).

3.5 Reconocimiento de tramas en la capa de transporte

En la capa de transporte, se encuentran protocolos basados en Ipv4, estos son TCP y UDP. Dado que están basados en IPv4, si ha sido reconocida una trama IPv4, se entra en un nuevo "case", para saber si se trata de una trama TCP o UDP. El reconocimiento de TCP, se encuentra en la línea 321 del Listado A-1 y el de UDP en la línea 342 del mismo listado. "protocolo_UDP" y "protocolo_TCP" son constantes que identifican a cada protocolo y están definidas en "TCP__CTE", la variable "protocolo", es de tipo Word y se ubica en el octeto número 17 de la trama actual. Si "protocolo" es igual a "protocolo_TCP", se trata de una trama TCP, por el contrario, si es igual a "protocolo_UDP", se trata de una trama UDP. Luego, a partir de la fila 13 y en la columna 1, se llena la pantalla con los nombres de los campos correspondientes a cada protocolo. A partir de la misma fila 13, pero en la columna 2, se llenan los valores numéricos de cada campo.

La Figura 3-5 muestra los campos del protocol UDP y la Figura 3-6 los campos del protocolo TCP. Nótese que los campos son ubicados inmediatamente debajo de los campos correspondientes a IPv4, con el fin de abarcar todos los datos de la trama.

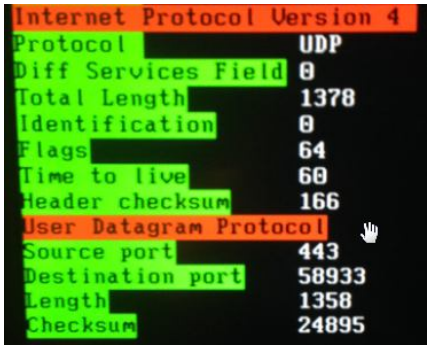

Figura 3-5 Trama UDP (fuente: propia).

Figura 3-6 Trama TCP (fuente: propia).

3.6 Reconocimiento trama ICMP

Este protocolo ubicado en la capa de red del modelo OSI, también está basado en IPv4, por lo cual se ubica en un case dentro del reconocimiento IPv4, junto a los protocolos TCP y UDP. El reconocimiento y llenado de pantalla con los datos se aprecia en el Listado A-1, desde la línea 302 hasta la 318.

El funcionamiento del programa es similar a los protocolos antes descritos. El resultado en pantalla se muestra en la Figura 3-7.

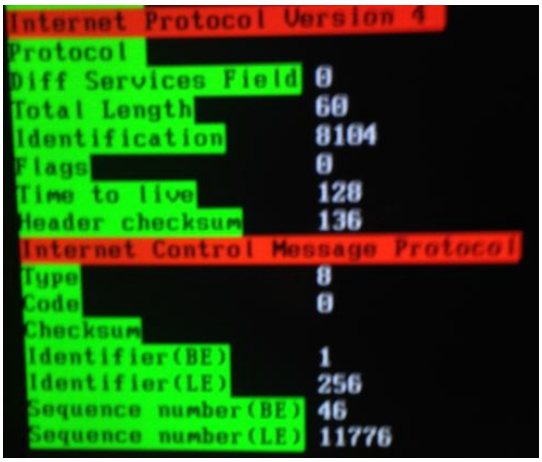

Figura 3-7 Trama ICMP (fuente: propia).

3.7 Reconocimiento trama ARP

El protocolo ARP, ubicado en la capa de transporte, tiene un identificador distinto al de IPv4, pero ubicado en el mismo sitio de la trama, por lo cual se debe preguntar por el en un "case" a parte del IPv4. Está ubicado desde la línea 401 del Listado A-1, hasta la 419. El reconocimiento y llenado de campos es similar a los protocolos antes descritos.

En la Figura 3-8, se aprecia el resultado en pantalla.

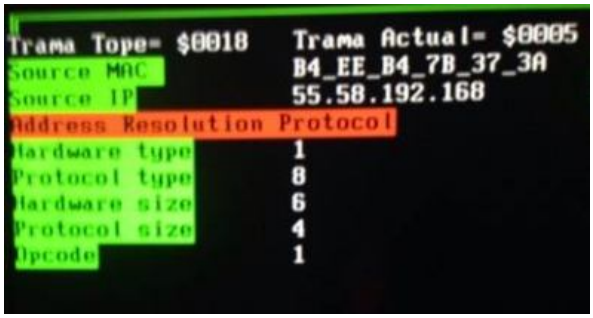

Figura 3-8 Trama ARP (fuente: propia).

3.8 Reconocimiento trama DNS

Ubicada en la capa de aplicación, la trama DNS generalmente es transmitida mediante un header UDP, por lo cual se debe hacer la consulta dentro de un "case" luego de haber reconocido la trama UDP. Este proceso se encuentra a partir de la línea 358 del Listado A-1, hasta la línea 390. Se hace la consulta por los dos casos más comunes de trama DNS, se crearon dos constantes: "DNS_QUERY" y "DNS_Q_RESPONSE", las cuales se encuentran definidas en TCP__CTE. Si la variable "DNS_FLAGS" es igual a "DNS_QUERY", entonces se trata de una trama de petición, por el contrario, si es igual a "DNS_Q_RESPONSE", se trata de una trama de respuesta. En ambos casos, una vez reconocido el tipo de trama, se procede a hacer el llenado de la pantalla con los campos correspondientes al protocolo, como se aprecia en la Figura 3-9.

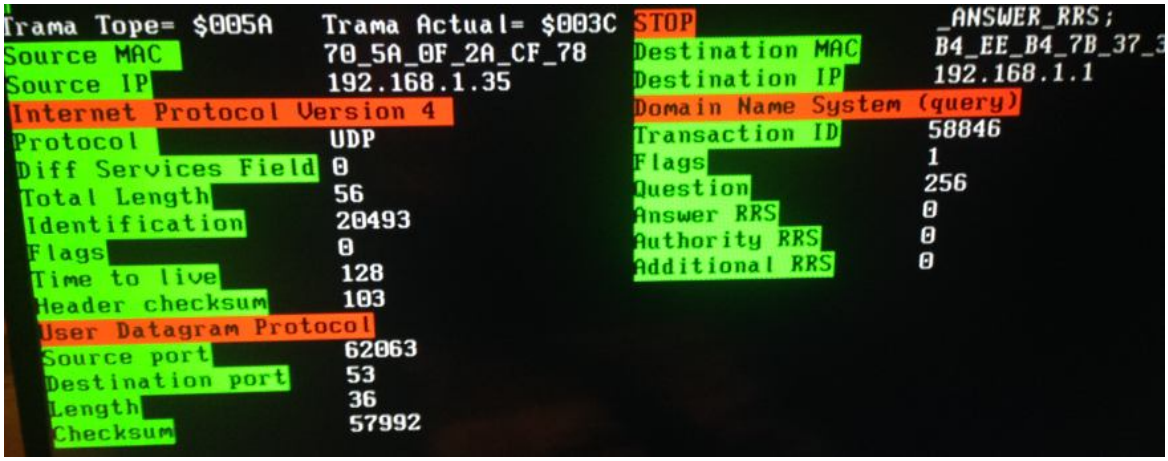

Figura 3-9 Domain Name System (query) (fuente: propia).

3.9 Reconocimiento de tramas extra

Además de los protocolos ya mencionados, a partir de la línea 420 hasta la 431 del Listado A-1. Se encuentra el reconocimiento de tramas del tipo IPv6 y protocolo RARP. Estas opciones sólo mostrarán el tipo de protocolo, sin visibilidad de sus campos. Además, se agrega un último case, en caso de no ser reconocido el protocolo, de esta forma, se mostrará en pantalla “UNKNOWN”.

Discusión y conclusiones

El estudio y desarrollo del sniffer, ha permitido comprender cómo se transmiten los datos y cómo se comportan distintos protocolos dentro de una red. El desarrollo de esta herramienta, entrega una interfaz de fácil uso y amigable a la vista, por lo que, además de servir para encontrar y solucionar problemas en una red, puede ser utilizado para fines educativos.

Lograr el desarrollo de un sniffer que es indetectable, implica una gran herramienta de defensa y monitoreo para el uso de empresas o redes que necesitan estar protegidas.

Lamentablemente, debido a los imprevistos en la estructura de DANUX, fue posible comenzar la programación recién en el mes de diciembre, por lo cual se limitó bastante el tiempo de trabajo, lo que no permitió agregar más protocolos al Sniffer. Esto genera una desventaja con respecto a otros sniffers, dado la cantidad de protocolos limitados que reconoce.

La puesta en pantalla de las tramas con sus campos respectivos, se puede apreciar de forma clara y de manera bastante amigable para el usuario, además de mostrar información fidedigna (esto comprobado comparando los valores con el sniffer Wireshark. Es importante destacar que los principales protocolos del estándar Ethernet son reconocidos por el sniffer, por lo cual es una gran herramienta para el análisis de tramas, con la ventaja de que no puede ser detectado.

En las tramas TCP, los campos de “sequence number” y “acknowledgement number” son de tipo doble palabra y vienen en formato “little endian”, por lo cual muestran un dato erróneo en pantalla. Se debe crear una estructura de datos, que invierta el orden de estos campos, para que, al mostrarlos en pantalla, esto sea en formato “big endian”.

El sniffer está desarrollado pensando en tramas IPv4, sin embargo, debido a la gran cantidad de dispositivos que trabajan sobre ethernet en el mundo, el rango de direccionamiento de IPv4 es actualmente limitado. Esto conlleva a la inminente migración a IPv6. Será necesario en un futuro próximo, crear las estructuras necesarias que reconozcan los dispositivos de tipo IPv6, además de desarrollar un nuevo código que permita monitorear los protocolos basados en IPv6.

Queda para futuro agregar de manera paulatina diversos protocolos de comunicación de las capas superiores del modelo OSI, a fin de equiparar las prestaciones de los demás analizadores en el mercado.

Bibliografía

[1] G. G. Castillo, «http://mixteco.utm.mx,» 2005. [En línea]. Available: http://mixteco.utm.mx/~resdi/historial/materias/IPv4.pdf.

[2] «Wireshark,» [En línea]. Available: https://www.wireshark.org/#download.

[3] «Acrylic,» [En línea]. Available: https://www.acrylicwifi.com/software/escaner-wifi-acrylic-wifi-gratuito/descargar-escaner-wifi-gratis-para-windows/.

[4] «Ettercap,» [En línea]. Available: https://ettercap.github.io/ettercap/downloads.html.

[5] «Kismet,» [En línea]. Available: https://www.kismetwireless.net/download.shtml.

[6] «Unicen,» [En línea]. Available: http://exa.unicen.edu.ar/catedras/comdat1/material/ElmodeloOSI.pdf.

[7] «RFC,» Septiembre 1981. [En línea]. Available: http://www.rfc-es.org/rfc/rfc0793-es.txt..

[8] J. L. G. Sánchez, «Gitaca,» [En línea]. Available: http://www.gitaca.es/jlgs/Docen/Transpas-TCP.pdf.

[9] «RFC,» Septiembre 1981. [En línea]. Available: http://www.rfc-es.org/rfc/rfc0793-es.txt.

[10] R. S. Corp, «Realtek Gigabit Ethernet media access controller with power management RTL8169». Taiwan Patente 1.21, 2002.

A Código programa principal

```
1   Program Sniff_50;
2      {***************************************}
3      {*        SNIFFER                      *}
4      {***************************************}
5   Const
6     { tamanno maximo de las tramas }
7     max_size         = 2016;{ maximo tamanmo  de las tramas }
8
9     null             = $00;
10    press_Contr      = $1D;
11    press_Shift      = $2A;
12    press_ALt        = $38;
13    reles_Contr      = $9D;
14    reles_Shift      = $AA;
15    reles_ALt        = $B8;
16    Bloq_May         = $BA;
17    Bloq_Num         = $C5;
18    Rel_Bloq_may     = $BA;
19    cursor_le        = $4B;
20    cursor_Ri        = $4D;
21    cursor_up        = $48;
22    cursor_do        = $50;
23    page_up          = $51;
24    page_down        = $49;
25    inicio           = $47;
26    fin              = $4F;
27    esc              = $01;
28    suprimir         = $53;
29    insertar         = $52;
30    backspace        = $0E;
31    return           = $0D;
32    F1               = $3B;
33    F2               = $3C;
34    F3               = $3D;
35    F4               = $3E;
36    F5               = $3F;
37    F6               = $40;
38    F7               = $41;
39    F8               = $42;
40    F9               = $43;
41    F10              = $44;
42    F11              = $57;
43    F12              = $58;
44    pausa            = $1D;
45    control_y        = $19;
46    Led_Cap_Lock     = $04;
47    Led_Num_Lock     = $02;
48    Led_Scroll_Lock  = $01;
49
50    {********** MEMORIA BUFFER ******************}
51    { Zona de memoria usada como Buffer          }
52    { se define una base_buffer                  }
```

```
53    buffer_base = $00F00000 ; { 16 Megas }
54
55    {********** MEMORIA DE VIDEO ****************}
56    { Zona de memoria de la pantalla              }
57
58  Var
59        MacF     : Mac ;
60        MacD     : Mac ;
61
62        IpF      : IPv4 ;
63        IpD      : IPv4 ;
64
65        {*********************************************}
66        {** variables usadas por el dump de sniffer   **}
67        {*********************************************}
68
69         {*********************************************}
70         {*          Contador de tramas               *}
71         {*********************************************}
72         { Contador de tramas capturadas               }
73         trama_count    : Word  ;
74
75         { Contador de tramas capturadas, en formato   }
76         { Word, se usa con dump_hexa                  }
77         top_out : Word                  Mem [$B80BA] ;
78         act_out : Word                  Mem [$b80E6] ;
79
80         {*********************************************}
81         {*           Trama actual                    *}
82         {*********************************************}
83         { Numero de trama actual en pantalla          }
84         trama_actual   : Word ;
85
86         {*********************************************}
87         {*           Trama size                      *}
88         {*********************************************}
89         { Numero de trama actual en pantalla          }
90         { debe ser tipo Mem, por dump_hexa            }
91         trama_size_L : Byte Mem [buffer_base+4  ];
92         trama_size_H : Byte Mem [buffer_base+5  ];
93         trama_size   : Word Absolute trama_size_L;
94
95         { Tramas actual en formato palabra con addr   }
96         { base en pantalla se usa para colocar el     }
97         { mensaje: "trama actual = "                  }
98         head_size : parrafo [12] mem [$B802A];
99
100        { Numero de trama actual, en formato  Word    }
101        { se usa con dump_hexa                        }
102        size         : Word;
103
104        dummy_0      : Array [1..20000] of byte;
105        scan         : Byte ;
106
107        run          : Boolean ;
108        pantalla_alt : Boolean ;
109
110        base         : Word    ;
111
112  Const
113
114      SERVER_DHCP  = FALSE;
115      SERVER_SIP   = FALSE;
116
117      RTL_1 = $CC00;
118
119      #INCLUDE  <RTL_8169.TXT>
120      #INCLUDE  <TCP__CTE.TXT>
121      #INCLUDE  <SIP__CTE.TXT>
122
123     {Configuraciones para DHCP}
```

```
124     DHCP_Net_IP              = 192.168. 100.  0;
125     DHCP_Brdcst_addr         = 192.168.  0.255;
126     DHCP_SUBNET_MASK_        = 255.255.255.  0;
127     DHCP_ROUTER_             = 192.168.100.  1;
128     DHCP_DOMAIN_NAME_SERVER_ = 192.168.100  .1;
129     DHCP_DOMAIN_NAME_        = 'Labproce';
130     DHCP_HOST_NAME_          = 'Danux';
131     DHCP_LEASE_TIME          = 43200; {Prestamo de IP: 12 horas}
132     DHCP_REBINDING_TIME      = 36000; {Revinculacion : 10 horas}
133     DHCP_RENEWAL_TIME        = 21600; {Renovacion    :  6 horas}
134
135     {Tipos de mensaje ARP}
136     ARP_REQUEST   = 1;
137     ARP_REPLY     = 2;
138
139     {Tipos de mensaje ICMP}
140     ICMP_ECHO_REQUEST = $08;
141     ICMP_ECHO_REPLY   = $00;
142
143     {Flags TCP}
144     TCP_SYN      = $02;
145     TCP_ACK      = $10;
146     TCP_SIN_ACK  = $11;
147     TCP_SYN_ACK  = $12;
148     TCP_PSH_ACK  = $18;
149
150     RX_Max  = 800;
151     TX_Max  =  16;
152
153 Type
154     mensa_VGA = text_vga [20];
155     #INCLUDE <TCP__TYP.TXT>
156
157
158 Var
159    vga :  Array [1..25] Of Array [1..4] of mensa_vga Mem [$B8000];
160
161    #INCLUDE  <TCP__VAR.TXT>
162
163 Produccion limpia_pantalla ;
164  Desde
165   Fill (VGA[3],00,4000) ;
166
167   VGA [2][1]   := Char ($07)+ 'Trama Tope= $' ;
168   Dump_Word (top_out,trama_count);
169
170   VGA [2][2]   := Char ($07)+ 'Trama Actual= $' ;
171   Dump_Word (act_out,trama_actual);
172
173   VGA [3][1] := ' Source MAC ';
174   macF       :=  trama_in [trama_actual] | SOU_mac ;
175   VGA [3][2] :=  char ($07)+#MacF;
176
177   VGA [3][3] := ' Destination MAC';
178   macD     :=  trama_in [trama_actual] | DES_Mac ;
179   VGA [3][4] :=  char ($07)+#MacD;
180
181   VGA [4][1] := ' Source IP';
182   ipF      := trama_in [trama_actual] | SOU_IP ;
183   VGA [4][2] :=   char ($07)+#ipF;
184
185   VGA [4][3] := ' Destination IP';
186   ipD      := trama_in [trama_actual] | DES_IP ;
187   VGA [4][4] :=   char ($07)+#IpD;
188
189 Hasta;
190
191
192 Produccion Dump_Alt ;
193 var
194  x : Byte;
```

```
 parraf_frame : Word;
 residuo      : Word;
 head_type    : Word ;
 Desde
   limpia_pantalla;
   fill (VGA [5],$00,160);

   size := trama_in [trama_actual] | frame_length ;
   If size-base > $140
    Then size := $140;
   Dump (VGA [6],trama_in [trama_actual],base,size);
 Hasta ;

Produccion trama_ICMP;
 Desde
   VGA [14][1] := ' Type';
   VGA [15][1] := ' Code';
   VGA [16][1] := ' Checksum';
   VGA [17][1] := ' Identifier(BE)';
   VGA [18][1] := ' Identifier(LE)';
   VGA [19][1] := ' Sequence number(BE)';
   VGA [20][1] := ' Sequence number(LE)';
 Hasta;

Produccion trama_ARP;
 Desde
   VGA [7][1]  := ' Hardware type';
   VGA [8][1]  := ' Protocol type';
   VGA [9][1]  := ' Hardware size';
   VGA [10][1] := ' Protocol size';
   VGA [11][1] := ' Opcode';
 Hasta;

Produccion trama_DNS;
 Desde
   VGA [6][3] := ' Transaction ID';
   VGA [7][3] := ' Flags';
   VGA [8][3] := ' Question';
   VGA [9][3] := ' Answer RRS';
   VGA [10][3] := ' Authority RRS';
   VGA [11][3] := ' Additional RRS';
 Hasta;

Produccion trama_TCP;
 Desde
   VGA [14][1] := ' Source Port';
   VGA [15][1] := ' Destination Port';
   VGA [16][1] := ' Sequence Number';
   VGA [17][1] := ' Acknowledgment Number';
   VGA [18][1] := ' Flags';
   VGA [19][1] := ' Window Size';
   VGA [20][1] := ' Checksum';
   VGA [21][1] := ' Urgent Pointer';
 Hasta;


Produccion
Dump_trama ;
var
 x : Byte;
 parraf_frame : Word;
 residuo      : Word;
 head_type    : Drow;

 Desde
   ? pantalla_alt= true ;
     Dump_alt ;
  Sino
     head_type := trama_in [trama_actual] | tipo;
     Desde
          ? head_type = protocolo_IP;
```

```
266              limpia_pantalla;
267
268                VGA [5][1]  := '@Internet Protocol Version 4 ';
269                VGA [6][1]  := ' Protocol ';
270                VGA [7][1]  := ' Diff Services Field';
271                VGA [8][1]  := ' Total Length';
272                VGA [9][1]  := ' Identification';
273                VGA [10][1] := ' Flags';
274                VGA [11][1] := ' Time to live';
275                VGA [12][1] := ' Header checksum';
276
277                SERV := trama_in [trama_actual] | Servicio;
278                VGA [7][2] := char ($07)+#SERV;
279
280
281                LARG := trama_in [trama_actual] | Largo;
282                VGA [8][2] := char ($07)+#larg;
283
284                IDENT := trama_in [trama_actual] | Identif;
285                VGA [9][2] := char ($07)+#IDENT;
286
287                FLGS := trama_in [trama_actual] | Flags;
288                VGA [10][2] := char ($07)+#FLGS;
289
290                TIMETL := trama_in [trama_actual] | TTL;
291                VGA [11][2] := char ($07)+#TIMETL;
292
293                CHECK := trama_in [trama_actual] | Checksum;
294                VGA [12][2] := char ($07)+#CHECK;
295
296
297
298               { *************************}
299               { analisis de protocolos IP }
300               { *************************}
301               Desde
302                    ? trama_in [trama_actual] | protocolo =protocolo_ICMP;
303                      trama_ICMP;
304                      VGA [13][1] := '@Internet Control Message Protocol';
305                      TYPE_ICMP  := trama_in [trama_actual] | ICMP_TYPE;
306                      VGA [14][2]:= char ($07)+#TYPE_ICMP;
307                      CODE_ICMP  := trama_in [trama_actual] | ICMP_CODE;
308                      VGA [15][2]:= char ($07)+#CODE_ICMP;
309                      CHECKSUM_ICMP:= trama_in [trama_actual] | ICMP_CHECKSUM;
310                      Dump_word (MEM[@VGA[16][2]],CHECKSUM_ICMP);
311                      ID_ICMP    := trama_in [trama_actual] | ICMP_ID_BE;
312                      VGA [17][2]:= char ($07)+#ID_ICMP;
313                      ID_ICMP    := trama_in [trama_actual] | ICMP_ID_LE;
314                      VGA [18][2]:= char ($07)+#ID_ICMP;
315                      SEQN_ICMP  := trama_in [trama_actual] | ICMP_SEQN_BE;
316                      VGA [19][2]:= char ($07)+#SEQN_ICMP;
317                      SEQN_ICMP  := trama_in [trama_actual] | ICMP_SEQN_LE;
318                      VGA [20][2]:= char ($07)+#SEQN_ICMP;
319
320
321                Sino
322                    ? trama_in [trama_actual] | protocolo =protocolo_TCP;
323                      VGA [6][2] :=  char ($07)+ 'TCP ';
324                      VGA [13][1]:= '@Transmission Control Protocol';
325                      trama_TCP;
326                      SPORT_TCP  := trama_in [trama_actual] | SOU_PORT;
327                      VGA [14][2]:= char ($07)+#SPORT_TCP;
328                      DPORT_TCP  := trama_in [trama_actual] | DES_PORT;
329                      VGA [15][2]:= char ($07)+#DPORT_TCP;
330                      SEQNUM_TCP := trama_in [trama_actual] | TCP_SEQ_NUM;
331                      VGA [16][2]:= char ($07)+#SEQNUM_TCP;
332                      ACKNUM_TCP := trama_in [trama_actual] | TCP_ACK_NUM;
333                      VGA [17][2]:= char ($07)+#ACKNUM_TCP;
334                      FLAGS_TCP  := trama_in [trama_actual] | TCP_FLAGS;
335                      VGA [18][2]:= char ($07)+#FLAGS_TCP;
336                      WSIZE_TCP  := trama_in [trama_actual] | TCP_WIN_SIZE;
```

```
337                     VGA [19][2]:= char ($07)+#WSIZE_TCP;
338                     CHK_TCP    := trama_in [trama_actual] | TCP_CKSUM;
339                     VGA [20][2]:= char ($07)+#CHK_TCP;
340                     UPTR_TCP   := trama_in [trama_actual] | TCP_URG_PTR;
341                     VGA [21][2]:= char ($07)+#UPTR_TCP;
342                Sino
343                    ? trama_in [trama_actual] | protocolo =protocolo_UDP;
344                     VGA [6][2] :=  char ($07)+ 'UDP ';
345                     VGA [13][1]:= '@User Datagram Protocol';
346                     VGA [14][1]:= ' Source port';
347                     VGA [15][1]:= ' Destination port';
348                     VGA [16][1]:= ' Length';
349                     VGA [17][1]:= ' Checksum';
350                   SOURCE_UDP     := trama_in [trama_actual] | SOU_PORT;
351                   VGA [14][2]    := char ($07)+#SOURCE_UDP;
352                   DESTINATION_UDP:= trama_in [trama_actual] | DES_PORT;
353                   VGA [15][2]    := char ($07)+#DESTINATION_UDP;
354                   LENGTH_UDP     := trama_in [trama_actual] | largo_udp;
355                   VGA [16][2]    := char ($07)+#LENGTH_UDP;
356                   CHK_UDP        := trama_in [trama_actual] | checksum_UDP;
357                   VGA [17][2]    := char ($07)+#CHK_UDP;
358            FLAGS_DNS := trama_in [trama_actual] | DNS_FLAGS;
359              Desde
360               ? trama_in [trama_actual] | DNS_FLAGS = DNS_QUERY;
361                 VGA [5][3] := '@Domain Name System (query)';
362                 trama_DNS;
363                 T_ID_DNS    := trama_in [trama_actual] | DNS_TRANSACTION_ID;
364                 VGA [6][4] := char ($07)+#T_ID_DNS;
365                 FLAGS_DNS   := trama_in [trama_actual] | DNS_FLAGS;
366                 VGA [7][4] := char ($07)+#FLAGS_DNS;
367                 Q_DNS       := trama_in [trama_actual] | DNS_QUESTION;
368                 VGA [8][4] := char ($07)+#Q_DNS;
369                 A_DNS       := trama_in [trama_actual] | DNS_ANSWER_RRS;
370                 VGA [9][4] := char ($07)+#A_DNS;
371                 AUTH_DNS    := trama_in [trama_actual] | DNS_AUTHORITY_RRS;
372                 VGA [10][4] := char ($07)+#AUTH_DNS;
373                 ADD_DNS     := trama_in [trama_actual] | DNS_ADDITIONAL_RRS;
374                 VGA [11][4] := char ($07)+#ADD_DNS;
375              Sino
376               ? trama_in [trama_actual] | DNS_FLAGS = DNS_Q_RESPONSE;
377                 VGA [5][3] := '@Domain Name System (response)';
378                 trama_DNS;
379                 T_ID_DNS    := trama_in [trama_actual] | DNS_TRANSACTION_ID;
380                 VGA [6][4] := char ($07)+#T_ID_DNS;
381                 FLAGS_DNS   := trama_in [trama_actual] | DNS_FLAGS;
382                 VGA [7][4] := char ($07)+#FLAGS_DNS;
383                 Q_DNS       := trama_in [trama_actual] | DNS_QUESTION;
384                 VGA [8][4] := char ($07)+#Q_DNS;
385                 A_DNS       := trama_in [trama_actual] | DNS_ANSWER_RRS;
386                 VGA [9][4] := char ($07)+#A_DNS;
387                 AUTH_DNS    := trama_in [trama_actual] | DNS_AUTHORITY_RRS;
388                 VGA [10][4] := char ($07)+#AUTH_DNS;
389                 ADD_DNS     := trama_in [trama_actual] | DNS_ADDITIONAL_RRS;
390                 VGA [11][4] := char ($07)+#ADD_DNS;
391              Sino
392
393              Hasta;
394
395
396
397
398                Sino
399                     VGA [5][2] :=  char ($07)+ '? ';
400              Hasta;
401        Sino
402            ? head_type = protocolo_ARP;
403              limpia_pantalla;
404              VGA [5][1] := '@Address Resolution Protocol';
405              VGA [6][1] := ' Hardware type';
406              VGA [7][1] := ' Protocol type';
407              VGA [8][1] := ' Hardware size';
```

```
408             VGA [9][1] := ' Protocol size';
409             VGA [10][1]:= ' Opcode';
410             HW_type    := trama_in [trama_actual] | ARP_HARDWARE_TYPE;
411             VGA [6][2] := char ($07)+#HW_type;
412             Proto_type := trama_in [trama_actual] | ARP_PROTOCOL_TYPE;
413             VGA [7][2] := char ($07)+#Proto_type;
414             HW_size    := trama_in [trama_actual] | ARP_HARDWARE_SIZE;
415             VGA [8][2] := char ($07)+#HW_size;
416             Proto_size := trama_in [trama_actual] | ARP_PROTOCOL_SIZE;
417             VGA [9][2] := char ($07)+#Proto_size;
418             Opcode     := trama_in [trama_actual] | ARP_OPCODE;
419             VGA [10][2]:= char ($07)+#Opcode;
420       Sino
421           ? head_type = protocolo_RARP;
422             limpia_pantalla;
423             VGA [5][2] :=  char ($07)+ 'RARP  ';
424       Sino
425           ? head_type = protocolo_IPv6;
426             limpia_pantalla;
427             VGA [5][1] := '@Internet Protocol Version 6';
428       Sino
429             limpia_pantalla;
430             VGA [5][2] :=  char ($07)+ 'UNKNOWN  ';
431      Hasta;
432  Hasta ;
433
434
435 Procedure Init_NICs;
436
437  Var
438     NIC_Cnt      : Word;
439     salida       : Byte;
440     nic_number   : Word;
441     test         : Byte;
442     diff         : Byte;
443     val32        : Longint;
444     val32aux     : integer;
445     val8         : Integer;
446     cha          : Char;
447     option       : Integer;
448     {rx set mode}
449     mc_filter    : Array [0..2] of Longint;
450     rx_mode      : Integer;
451     aux_byte     : Byte;
452     cplus        : word;
453     tmp          : word;
454     cnt          : Integer;
455     aux          :byte;
456
457  Begin  { Init_NIC  }
458     For NIC_Cnt := 1 to  max_NIC DO  ;
459        Begin
460
461         Command := RTL_NIC_Reset;
462         While Command = RTL_NIC_Reset Do;
463
464         Cfg9346  := Cfg9346_Unlock;
465         Config1  := Config1 Or Power_Manager;
466         Config2  := PCI_Bus_32_Bits;
467         Config5  := Enable_Reset_PCI;
468         Cfg9346  := Cfg9346_Lock;
469
470
471         IntrStatus := $FFFF; {reseteo de todos los bits}
472
473         (*********************************************************)
474         (**  Determina El tipo de Chips  de la NIC               **)
475         (*********************************************************)
476         val32 := Tx_Config And $7E000000; { HWVERID0 }
477         aux_byte:= val32 shr 25;
478         NIC[NIC_Cnt].nic_name:= aux_byte;
```

```
479
480         NIC [NIC_CNT].pcfg := PCFG_METHOD_3;
481
482         (*********************************************************)
483         (**  Configuracion de medio fisico de tarjeta NIC_Count   **)
484         (*********************************************************)
485         {   Case NIC[NIC_Cnt].nic_name Of
486         End;}
487
488         Port[base_nic + $82] := $01;
489         (*********************************************************)
490         (**  El registro $82 es reservado: no tengo idea que hace **)
491         (**  al parecer habilita la modificacion de parametros    **)
492         (**  como latencia PCI y offsets                          **)
493         (*********************************************************)
494
495         {   If (NIC[NIC_Cnt].mcfg = MCFG_METHOD_2)
496         Then PortW [NIC_Cnt,$0B , $0000);        }
497
498         {  Modo de Operacion: CONFIG REGISTER WRITE ENABLE }
499         Int_Status := $FFFF; {reseteo de todos los bits}
500         Cfg9346   := Cfg9346_Unlock;
501
502         {***************************** DMA SET ****************************}
503         {*  RDSAR: Receive Descriptor Start Address, OFFSET E4-EB          *}
504         {*  offset E7-E4: low 32 bit addr     Asignando direcciones DMA.   *}
505         {*  offset E8-EB: high 32 bit addr                                 *}
506         {*  Indicar buffers de descriptores a la tarjeta                   *}
507         {**************************************************************}
508         RxDescAddrHigh := $00000000;
509         RxDescAddrLow  := @descrip_RX;
510         TXdescAddrHigh := $00000000;
511         TXdescAddrLow  := @descrip_TX;
512
513         Writeln('***************** RX TX ENABLE! *****************');
514         Command := Cmd_Rx_Enb Or Cmd_Tx_Enb;
515         Early_Tx_Threshold := Threshlod_Level;
516
517         (*********************************************************)
518         (**                                                     **)
519         (** OJO  muy importante                                 **)
520         (**                                                     **)
521         (** Configura  max tamano de packet que puede leer      **)
522         (** debe der menor o igual que el tamano del desc RX    **)
523         (*********************************************************)
524         Rx_MaxSize := max_size;
525
526         (*********************************************************)
527         (**  Define  RxConfig, configuracion de la recepcion    **)
528         (*********************************************************)
529         Rx_Config:=  no_Theshold          Or
530                      RX_DMA_Burstno_limit Or
531                        Accept_Broadcast   Or
532                         Accept_My_Phys    Or
533                          Accept_multicast Or
534                            Promiscuous  ;
535
536         (*********************************************************)
537         (**  Define  TxConfig, configuracion de la transmision  **)
538         (*********************************************************)
539         Tx_Config := TX_DMA_Burst_1024_Bit Or
540                        Inter_Frame_Gap_0;
541
542         CCRC_Cmd := Litle_Indian_Mode+PCI_Multy_Rx_TX;
543
544         Rx_Missed_Packet :=0;
545
546         {- - - - - - - - - - - - - - - - - -  - - - - - - - - - - - - -}
547         {no early/rx INT}
548         Multi_Intr  := Multi_Intr And $F000;
549         IntrStatus  := 0;{ rtl8169_intr_mask }
```

```
550         writeln;
551        End; { for count }
552  End; { Init_NIC  }
553
554
555  Procedure Init_Buffer_NIC;
556   Begin
557      {********** Llenado de descriptores RX  **********}
558      For i := 1 to RX_Max  do
559          Begin
560          descrip_RX [i].status     := OWNbit Or MAX_SIZE;
561          descrip_RX [i].buff_haddr := $00;
562          descrip_RX [i].vlan_tag   := $0;
563          descrip_RX [i].buff_addr  := @trama_in [i];
564          End;
565      descrip_RX [RX_Max].status := descrip_RX [RX_Max].status Or
566                                      EORbit;
567      {********** Llenado de descriptores TX  **********}
568      For i := 1 to TX_Max  do
569          Begin
570          descrip_TX [i].status      := LSBIT OR fsbit Or MAX_SIZE;
571          descrip_TX [i].buff_haddr  := $00;
572          descrip_TX [i].vlan_tag    := $0;
573          descrip_TX [i].buff_addr   := @trama_out [i];
574          End;
575      descrip_TX [TX_Max].status := descrip_TX [TX_Max].status Or
576                                                      EORbit;
577
578   End; { Init_Buffer_NIC }
579
580
581
582  Procedure Stop_NIC;
583   Var
584      cnt        : longint;
585   Begin
586      For cnt:= 1 to max_NIC do
587       Begin
588        Command  :=   $00;
589        IntrMask := $0000;
590       End;
591   End;
592
593  Procedure Run_NIC ;
594  Begin
595   Command := Cmd_Rx_Enb Or Cmd_Tx_Enb ;
596  End;
597
598  Produccion capture_trama;
599  Desde
600   If RUN
601    Then
602     Begin
603       trama_count := trama_count + 1;
604       dump_word  (vga [2][2],trama_count);
605     End;
606  Hasta;
607
608  (*************************************************************************)
609  (************            Produccion  Principal                ************)
610  (*************************************************************************)
611  Desde
612      trama_actual := 1;
613      limpia_pantalla ;
614      writeln (my_mac);
615
616      { Iniciacion de variables y de la tarjeta *****}
617      max_NIC := 1;
618
619      Init_Buffer_NIC; {  descriptores RX TXN TXH y ubicacion de buffers }
620      Init_Nics;
```

```
621     Init_Nics;
622
623     i     := 1;
624     i_tx  := 0;
625     trama_count  := 0 ; { tramas capturadas }
626     trama_actual := 1 ;
627
628     RUN          := True;
629     VGA [2][3]   := ' RUN';
630     pantalla_alt := False ;
631     base         := $0000;
632
633     Repeat
634      scan := read_sca;
635      Desde
636          ? run = false ;
637            Desde
638                 ? (scan = page_up) And
639                   (trama_count >trama_actual) And (Not Run);
640                   trama_actual := trama_actual +1;
641                   base := $0000 ;
642                   dump_trama;
643             Sino;
644                  ? (scan = page_down) And (trama_actual >1) And (Not Run);
645                    trama_actual := trama_actual -1;
646                    base := $0000 ;
647                    dump_trama;
648              Sino
649                 ? (scan = inicio ) And (Not Run);
650                   trama_actual := 1 ;
651                   base := $0000 ;
652                   dump_trama;
653              Sino
654                 ? (scan = fin ) And (Not Run);
655                   trama_actual := trama_count ;
656                   base := $0000 ;
657                   dump_trama;
658              Sino
659                 ? (scan = Backspace) ;
660                   trama_actual  := 1 ;
661                   trama_count   := 0 ;
662              Sino
663                 ? (scan = cursor_le) And
664                   (trama_in [trama_actual]|frame_length >
665                    base+$0140+$0010) ;
666                   base := base+$0010 ;
667                   dump_trama ;
668              Sino
669                 ? (scan = cursor_ri) And
670                   (base >= $0010) ;
671                   base := base-$0010 ;
672                   dump_trama ;
673              Sino
674                 ? (scan = pausa);
675                   run := NOT RUN ;
676                   IF Run
677                    Then VGA [2][3] := ' RUN'
678                    Else VGA [2][3] := '@STOP';
679
680                   IF Not Run
681                    Then
682                     Begin
683                      Stop_nic  ;
684                      base := $0000 ;
685                      dump_trama ;
686                      scan := null;
687                     End
688                    Else
689                     Begin
690                      init_nics ;
691                      init_nics ;
```

```
692                     End;
693              Sino
694                  ? (scan = F1);
695                    pantalla_alt :=  Not pantalla_alt ;
696                    base := $0000 ;
697                    dump_trama ;
698              sino
699                  ? (scan =esc)
700
701              Sino { ojo es muy importante colocar esta opcioon vacia, }
702                    { sino declara que toda la opcion principal no      }
703                    { cumple y se sale del programa                     }
704                    if scan <> null
705                     Then writeln (scan,' ');
706                    scan:= null;
707             Hasta ;
708
709          Sino ;  {  Run = true  }
710            If ((descrip_RX [i].status and OWNbit) = 0)
711             Then
712              Begin
713               trama_count := i;
714               dump_word (top_out,trama_count);
715               size := (descrip_rx [i].status And $0FFF)-4;
716               Dump (VGA [3],trama_in [i],0,size);
717               trama_in [i] | frame_length := size;
718               If i <> RX_MAX
719                Then
720                 Begin
721                  while((descrip_RX[i].status and OWNbit) =0)
722                    do descrip_rx[i].status := $80000000 Or max_size;
723                  i := i + 1;
724                 End
725                Else
726                 Begin
727                   descrip_RX [Rx_Max].status := $C0000000 Or max_size;
728                   i := 1;
729                 End;
730              End ;
731       Hasta;
732       Desde
733            ? (scan = pausa);
734              run := NOT RUN ;
735              IF Run
736               Then VGA [2][3] := ' RUN'
737               Else VGA [2][3] := '@STOP';
738              scan := null;
739         Sino ;
740       Hasta;
741      Until scan = esc;
742  Hasta.
```