Alberto Campos Gordo

Analizador Gráfic niffer) en entorno GNU

Alberto Campos Gordo
ETIS
Consultora: Maria Isabel March i Hermo
21 de Enero del 2008

Dedico este TFC
a Maria del Carme, mi esposa,
a Héctor, mi hijo,
a Raquel, mi madre,
y en especial a Francisco, mi padre.

RESUMEN DEL TRABAJO

El presente Trabajo Final de Carrera se enmarca dentro del área de Redes de computadoras y consiste en la realización de una aplicación gráfica, en entorno GNU, que analice todo el tráfico de una red informática. Lo que se conoce como un Sniffer.

Para la realización de la aplicación, se pone en práctica conocimientos adquiridos a lo largo de la carrera en asignaturas como por ejemplo, "*Interacció Humana amb els Ordinadors*", "*Enginyeria del Programari*" y "*Fonaments de Programació II*" entre otras, así como un repaso general y profundo de la asignatura "*Xarxes de Computadors*", en espacial a lo relacionado con los Protocolos de Internet.

Se pretende desarrollar una herramienta orientada principalmente, que no exclusivamente, a fines educativos, cuyo propósito general es mostrar al usuario información detallada de cada uno de los protocolos capturados por la aplicación, ayudándole a comprender que se esconde detrás de cada uno de ellos.

Como se comenta en el párrafo anterior, no es descartable la utilización de la aplicación en el ámbito profesional para la detección de intrusos, monitorización y análisis del tráfico en una red de computadoras detectando los cuellos de botella y problemas que existan en ella.

El sniffer en cuestión se podrá ejecutar en cualquier máquina, ya que se emplea el lenguaje *JAVA* para su implementación. En general, una aplicación de Java tiene grandes posibilidades de ser ejecutable si modificación en cualquier computadora basada en las plataformas Apple Macintosh, Microsoft Windows o distintas versiones de Unix. Por esta razón se dice que el lenguaje Java es transportable.

La aplicación aunque es capaz de capturar cualquier clase de paquete, solo realiza el análisis de los siguientes protocolos:

- Tramas Ethernet
- Paquetes IP, ICMP y ARP
- Segmentos TCP y UDP

Para poder interactuar con ella se ha creado un entorno gráfico muy intuitivo para el usuario, de manera que su uso no requiere más que los conocimientos básicos en redes y protocolos.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

CAPÍTULO 1. INTRODUCCIÓN

1.1 Justificación del TFC y contexto en el cual se desarrolla: punto de partida y aportación del TFC

Este trabajo de final de carrera se enmarca dentro del área de Redes y su objetivo principal es realizar una herramienta grafica que permita analizar el tráfico de una red determinada.

Como punto de partida de este proyecto se realizan las siguientes acciones que puedan aportar las bases suficientes para empezar a trabajar:

- Se repasa los materiales de redes de computadores, destacando todo lo relacionado con los protocolos de Internet.
- Se repasa los materiales de otras asignaturas relacionadas con la elaboración del proyecto.
- Búsqueda en Internet de información relativa a otros sniffer ya existentes, así como información relativa al proyecto.

Después de lo anteriormente expuesto, se plantean cuestiones fundamentales para su realización; ¿son suficientes los conocimientos adquiridos a lo largo de la carrera para la elaboración del sniffer?, ¿se necesita ampliar en algún aspecto en concreto esos conocimientos?

Para la primera pregunta la respuesta es sí, analizando la envergadura del proyecto se necesita ampliar conocimientos en programación. Para la segunda pregunta la respuesta es sí, se necesita ampliar los conocimientos relativos a la creación de interfaces gráficas, ya que durante la carrera no se ha tratado este tema con la profundidad suficiente para realizar el proyecto.

Este TFC significa un reto a nivel personal, ya que no se tiene experiencia en el ámbito profesional y este proyecto por sí solo requiere una gran capacidad de trabajo y síntesis de todo lo asimilado a lo largo de estos años. Técnicamente, supone por un lado la aplicación de los conocimientos adquiridos en programación, redes y protocolos, y por otro la adquisición de nociones relativas a la creación de interfaces gráficas de usuario que posteriormente podrán ser aplicados en el campo profesional.

1.1.1 Ejemplos de aplicaciones similares en el mercado

Un sniffer es un programa para monitorizar y analizar el tráfico en una red de computadoras, detectando los cuellos de botella y problemas que existan. También puede ser utilizado para "captar", lícitamente o no, los datos que son transmitidos en la red.
Destacan por su importancia los siguientes: Tcpdump, Ngrep, Snot, Nwatch, Ethereal, Ettercap y Kismett. A modo de ejemplo se muestran dos analizadores de red citados anteriormente, Tcpdump y Ethereal, ya que ambos se han utilizado a lo largo de la carrera.

Tcpdump muestra las cabeceras de los paquetes que captura de un interfaz de red dado y que cumplen la expresión pasada al ejecutar, es decir, te permite monitorizar tráfico de red en tiempo real.

Uno de los inconvenientes de este sniffer es que funciona en línea de comandos, sin entorno grafico, no permitiendo el estudio exhaustivo de los paquetes capturados. Los filtros que se pueden crear para mostrar tan sólo la información que nos interesa,

hacen de tcpdump una herramienta muy potente para el análisis de tráfico en redes de comunicaciones. En Windows, en lugar del tcpdump se utiliza el Windump.

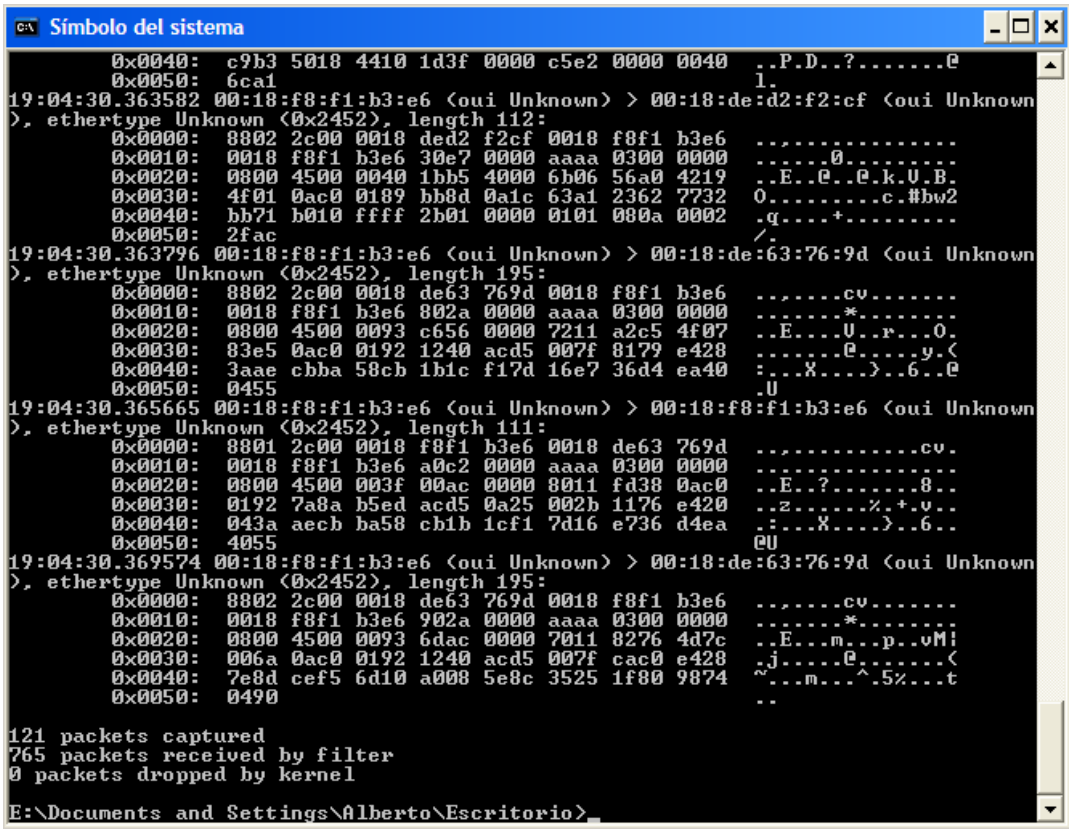


Figura 1.- Captura de tcpdump

Ethereal que ahora se llama Wireshark, es un potente analizador libre de protocolos de redes, para máquinas Unix y Windows. Nos permite capturar los datos directamente de una red u obtener la información a partir de una captura en disco (puede leer más de 20 tipos de formato distintos). Destaca también por su impresionante soporte de más de 300 protocolos.

Este sniffer trabaja sobre un entorno gráfico, a diferencia de Tcpdump, intuitivo y de fácil manejo que es toda una referencia en analizadores de protocolos. Sus múltiples funciones y sus detalladas capturas facilitan toda la información necesaria para analizar el funcionamiento de protocolos a nivel de red, a nivel de transporte y a nivel de aplicación.

Concretamente, con Ethereal se pueden capturar diálogos, listar tramas Ethernet, visualizar las cabeceras de cada trama y acceder a los datos adjuntos. Sus herramientas permiten filtrar las tramas por protocolo, por dirección física o por otros condicionales más complejos. En definitiva, es una excelente herramienta para ampliar conocimientos sobre protocolos en redes Ethernet.

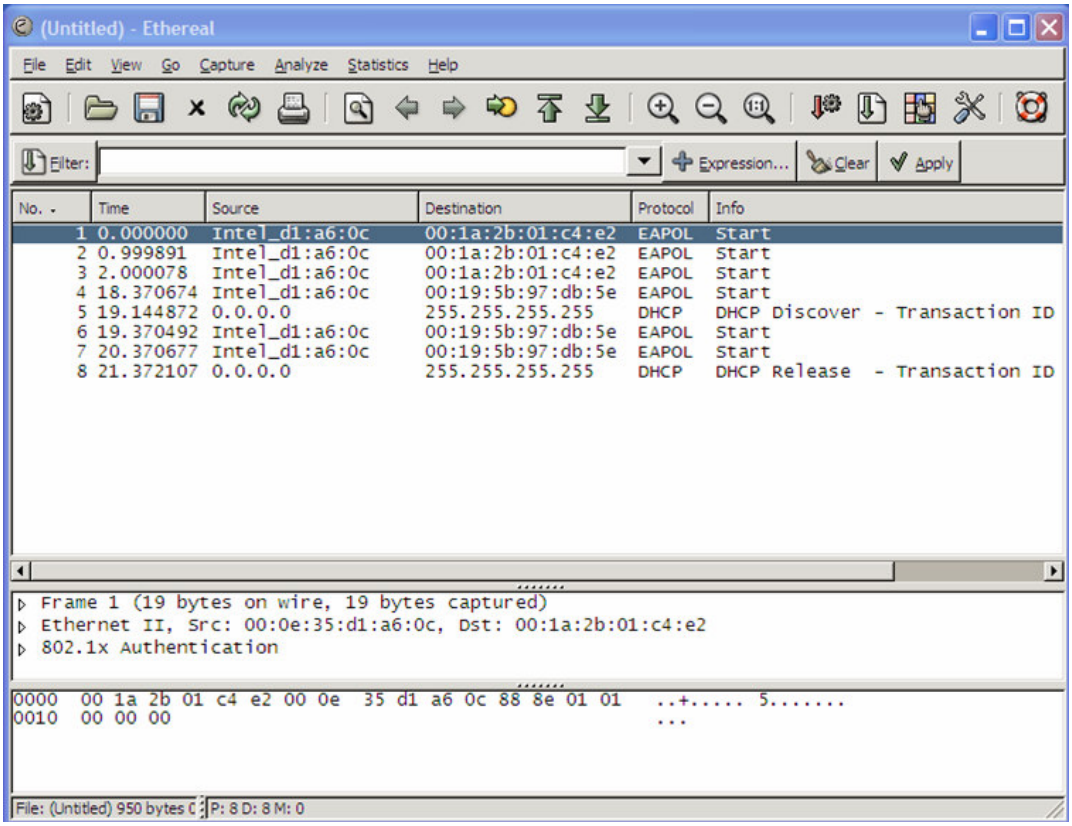


Figura 2.- Captura realizada con Ethereal

1.2 Objetivos del TFC

El objetivo final del TFC es la creación de una aplicación que capture y analice el tráfico de una red determinada. Para ello se propone dotar a la aplicación de las siguientes características y funcionalidades:

- Realizar capturas en tiempo real
- Soporte los siguientes protocolos: Ethernet, IPv4/v6, TCP, UDP, ICMP y ARP.
- Control estadístico de cada protocolo.
- Mostrar de alguna manera esos datos estadísticos.
- Dotar a la aplicación opciones de filtrado.
- Dotar a la aplicación de una Interfaz Grafica de Usuario (IGU), de fácil manejo y visualmente atractiva.

1.3 Planteamiento y método a seguir

Para el desarrollo del proyecto, basado en la programación orientada a objetos, se hará servir el método UML (Unified Modeling Language). El UML es un lenguaje grafico y flexible de modelado que permite describir modelos que representan sistemas basándose en los conceptos de la orientación a objetos.

En lo que se refiere al lenguaje de programación, se utiliza un lenguaje orientado a objetos, asociado como ningún otro a Internet y que permite la programación de

interfaces de usuario; este lenguaje es JAVA. Aparte de todo lo expuesto anteriormente, este lenguaje ha sido con el que se ha trabajado en la mayoría de las asignaturas de la UOC.

En la producción del sniffer se ha seguido un ciclo de vida iterativo e incremental, distinguiéndose las siguientes fases:

1) Investigación sobre los distintos productos similares en el mercado, así como realizar una valoración a nivel de dedicación y conocimientos que supone afrontar el proyecto.
2) Análisis y diseño del analizador de red, se estudia el dominio del problema, se establece la arquitectura general del programa y se realiza una planificación del proyecto
3) Implementación del analizador de manera iterativa e incremental, teniendo en cuenta todas las necesidades de información que ha de cumplir y se desarrolla la arquitectura obtenida de la fase anterior.
4) Fase de pruebas y redacción de la memoria.

1.4 Planificación del proyecto

Paso 1:

Tiempo: 2 semanas (del 30 de septiembre al 14 de octubre).
Descripción: recogida y clasificación de toda la información que pueda ser relevante para llevar a cabo el proyecto:

- Temas relacionados con los protocolos TCP/IP.
- Temas relacionados con los analizadores de red existentes.
- Temas relacionados con el análisis, el diseño y la implementación de sniffers.

Objetivos:

- Profundizar los conocimientos sobre los protocolos TCP/IP.
- Tener una visión clara sobre las funcionalidades que tendría el sniffer.
- Tener una visión general sobre algunos sniffers más usados en la actualidad.

Términos:

- Documento descriptivo de los protocolos TCP/IP.
- Informe sobre analizadores de red existentes.

Paso 2:

Tiempo: 4 semanas (del 15 de octubre al 11 de noviembre).
Descripción: análisis y diseño del analizador de red a implementar.
Objetivos:

- Establecer los requisitos y opciones que tendría que tener el sniffer.
- Determinar la técnica y el lenguaje de programación a utilizar.
- Obtener el análisis de programa en función de los requisitos y opciones que se le quiere dar.
- Obtener el diseño de la aplicación a implementar basándose en el análisis del punto anterior.

Términos:

- Obtener el análisis y el diseño de la aplicación documentando las decisiones tomadas.

Paso 3:

Tiempo: 6 semanas (del 11 de noviembre al 23 de diciembre).
Descripción: Implementación del sniffer.
Objetivos:
- Implementar el sniffer.
- Creación del sniffer.
- Documentación sobre la implementación.

Términos:
- Creación del sniffer.
- Documentación sobre la implementación.

Paso 4:

Tiempo: 1 semana (del 24 al 30 de diciembre)
Descripción: documentación del producto.
Objetivos:
- Documentar el uso de las diferentes opciones de configuración que pueda tener.

Términos:
- Documentación del analizador de red.

Paso 5:

Tiempo: 2 semanas (del 31 de diciembre al 13 de enero).
Descripción: revisión final de la memoria y preparar la presentación del analizador de red que se ha creado.
Objetivos:
- Síntesis de la memoria realizada durante el proyecto.
- Crear una presentación del analizador de red creado.

Términos:
- Memoria del proyecto.
- Presentación del proyecto.

Desviación temporal al plan de trabajo inicial

Este plan de trabajo inicial ha sufrido una desviación temporal causado por un pequeño reciclaje de los conocimientos en programación java y en especial en lo referente a la librería swing, de la que prácticamente no se tenía nociones suficientes para poder afrontar el proyecto con garantía de éxito. La desviación consiste en arañar una semana al paso 2 y añadirla al paso 3, es decir, la planificación quedaría de la siguiente manera:

Paso 2:
Tiempo: 3 semanas (del 15 de octubre al 4 de noviembre).

Paso 3:
Tiempo: 7 semanas (del 5 de noviembre al 23 de diciembre).

1.5 Productos obtenidos

Una vez finalizado el proyecto, en principio se habrán obtenidos los siguientes productos:

- Un analizar gráfico de red en entorno GNU, con las características ya comentadas anteriormente.
- La memoria del TFC.
- Manual de instalación y usuario (incluido en la memoria).

1.6 Breve descripción de los siguientes capítulos

Los siguientes capítulos están dedicados a cada una de las etapas de creación de la aplicación que se desarrollará, los cuales son:

- Análisis del proyecto (capítulo 2)
- Diseño del proyecto (capítulo 3)
- Implementación del proyecto (capítulo 4)
- Pruebas (capítulo 5)
- Manual de instalación y usuario (capítulo 6)

CAPÍTULO 2. ANÁLISIS DEL PROYECTO

2.1 Introducción

El análisis es la primera etapa del desarrollo del programa propiamente dicho, y que consiste en traducir los requisitos a una forma más adecuada para ser la base de partida de este proyecto.

Un primer cometido de el análisis es la de traducir los requisitos a un lenguaje más formal, que en el método que seguimos son los modelos y diagramas de UML. Un segundo cometido de la etapa de análisis será la identificación de unas clases fundamentales que serán la base de la implementación del programa. Un tercer cometido será la expresión de los casos de uso en relación a estas clases.

De la recogida y documentación de requisitos se ha extraído el siguiente modelo de dominio que recoge las clases más importantes:

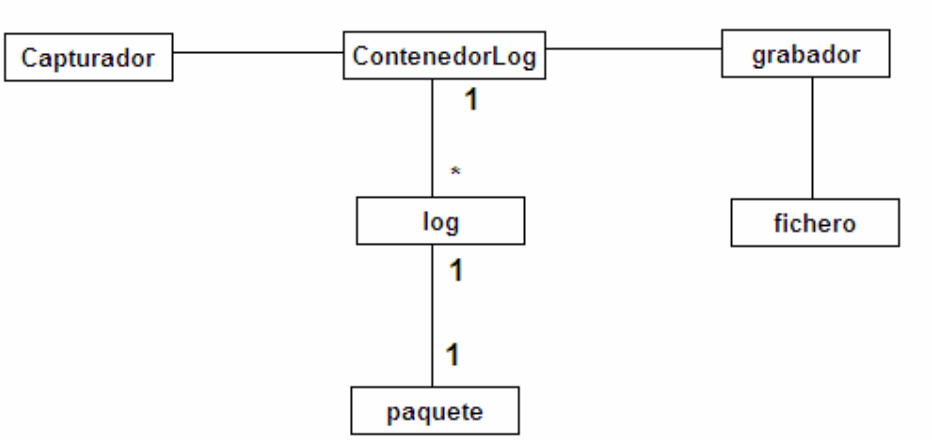


Figura 3.- Modelo de dominio

Donde la clase "paquete" encapsula los diferentes niveles de red, Internet (IP), transporte (TCP) y aplicación. En cada uno de los niveles de la red se encuentra protocolos diferentes. La situación relativa de cada protocolo en los diferentes niveles se muestra en la siguiente figura:

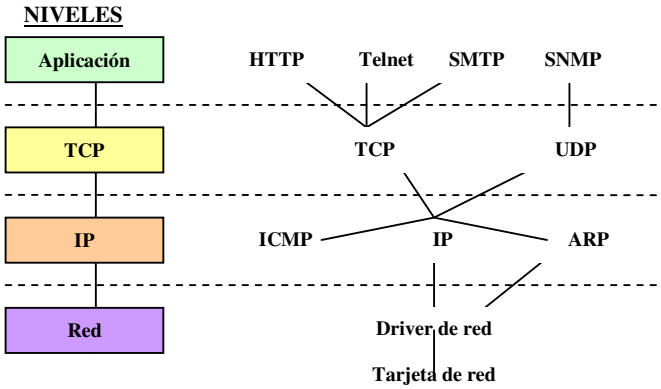


Figura 4.- Protocolos de Internet

El diagrama de casos de uso quedaría como se describe a continuación:

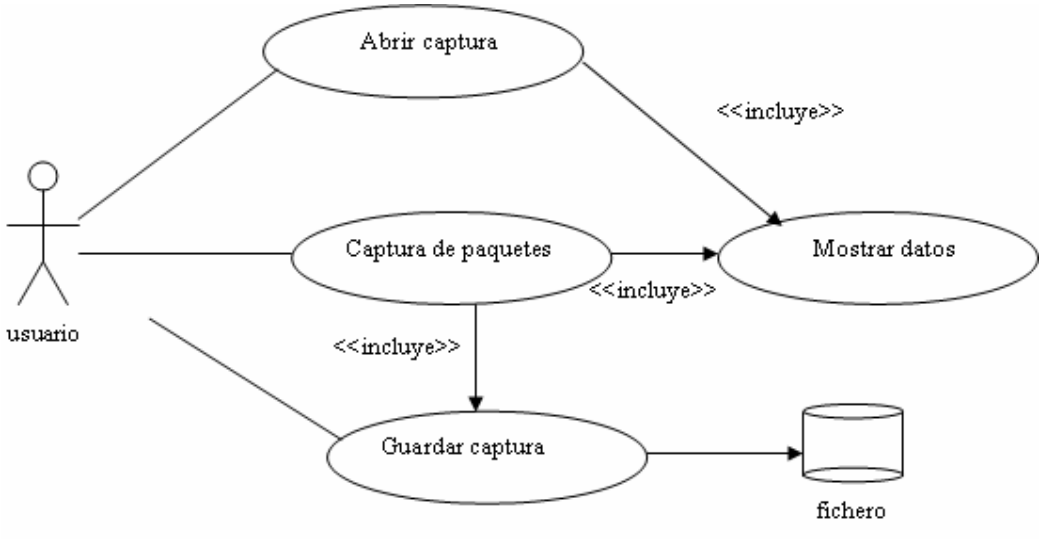


Figura 5.- Diagrama de casos de uso

2.2 Descripción de los casos de uso

Caso de uso número 1: "Mostrar datos"

Resumen de la funcionalidad: muestra por pantalla la captura de datos.

Actores: **usuario.**

Casos de uso relacionados: Abrir Captura, Captura de paquetes

Precondición: el fichero de captura existe o tenemos datos en el buffer de captura.

Poscondición: la pantalla nos muestra los datos de la captura.

El usuario después de abrir un fichero de captura existente o de finalizar una captura, puede ver por pantalla una lista detallada por campos de los paquetes capturados.

Alternativas de proceso y excepciones: la existencia algún problema con la apertura de un fichero o con la captura de los datos

Caso de uso número 2: "Captura de paquetes"

Resumen de la funcionalidad: permite una sesión de captura de paquetes que circulan por una tarjeta de red que previamente el usuario ha seleccionado.

Actores: **usuario.**

Casos de uso relacionados: Guardar Captura, Mostrar datos

Precondición: el usuario ha seleccionado la tarjeta de red con la que quiere realizar la captura.

Poscondición: obtenemos una captura que podemos visualizar por pantalla y posteriormente guardar en fichero.

El usuario selecciona la tarjeta de red por la cual quiere obtener la captura, así como otras opciones de filtrado. Los paquetes obtenidos de la captura se van mostrando por pantalla, cuando el usuario haga clic sobre el botón stop tendrá dos opciones, escoger entre guardarlos en un fichero o descartar su almacenamiento.

Alternativas de proceso y excepciones: la existencia algún problema con la creación de un fichero o con la captura de los datos.

Caso de uso número 3: "Guardar captura"

Resumen de la funcionalidad: Permite guardar una sesión de captura de paquetes en un fichero.

Actores: **usuario.**

Casos de uso relacionados: Captura de paquetes

Precondición: Tenemos datos en el buffer de captura.

Poscondición: Tenemos un fichero de la captura.

El usuario después de finalizar una captura, indica el nombre del fichero donde se va a almacenar los datos obtenidos de la sesión de captura y su ubicación en el sistema.

Alternativas de proceso y excepciones: la existencia algún problema con la apertura de un fichero.

Caso de uso número 4: "Abrir captura"

Resumen de la funcionalidad: recupera los datos de una captura ya existente almacenados previamente en un fichero.

Actores: **usuario.**

Casos de uso relacionados: Mostrar datos

Precondición: el fichero de captura existe.

Poscondición: la pantalla nos muestra los datos de la captura.

El usuario recupera una sesión de captura almacenada en un fichero y es mostrada por pantalla.

Alternativas de proceso y excepciones: la existencia algún problema con la apertura de un fichero.

CAPÍTULO 3. DISEÑO DEL PROYECTO

3.1 Introducción

La etapa de diseño hace de puente entre el análisis y la implantación. El modelo del análisis describe las funciones que ha de hacer la aplicación y también especifica los eventuales requisitos no funcionales, es decir, plantea el problema que se ha de resolver con la aplicación. La fase actual hace referencia con la elaboración de una solución de este problema. Por otro lado, el modelo de análisis no es una base adecuada para realizar la implementación directamente, ya que no expresa en términos de la tecnología que se aplicara en el proyecto (principalmente el lenguaje de programación y la herramienta de soporte de la interface grafica del usuario).

3.2 Diseño de los casos de uso

Caso de uso 1: "mostrar datos"

Una vez finalizada la captura en tiempo real o se ha recuperado una sesión anterior, el sistema muestra en una tabla los datos básicos de cada paquete capturado en la sesión.

1. Si el usuario hace click sobre una fila de la tabla, se obtiene información detallada del paquete hasta la capa de transporte.
2. Para cada nivel se crea una instancia de la clase del paquete correspondiente. Es decir, si la trama transporta un paquete UDP, se crea el paquete Ethernet, el paquete IP y el paquete UDP.
3. Cada clic sobre una fila implica la ejecución de todos los métodos de recuperación de datos que tienen los paquetes.
4. EL usuario puede salir de la visualización de los datos clickando en el botón salir, la cual cosa implica salir de la aplicación sin salvar la captura.

Caso de uso 2: "captura de paquetes"

El usuario visualiza la pantalla de opciones donde selecciona los parámetros que quiere aplicar a la captura. Una vez aprieta el botón "OK" la pantalla opciones desaparece y en su lugar se visualiza una nueva pantalla, con el número y tipo de paquetes que se van capturando.

Esta captura finaliza cuando el usuario hace click sobre el botón "stop" o cuando se llegue al número máximo permitido de paquetes capturados establecido (3000 paquetes).
La captura será guardada provisionalmente en un listado de paquetes, ya que se quiere que el programa interfiera lo menos posible en la actividad de la máquina.

Caso de uso 3: "guardar captura"

El usuario elige la opción del menú "guardar". Seguidamente visualiza una pantalla de navegación donde puede escoger la ubicación y el nombre del fichero a guardar. Una vez haga click en el botón "guardar", el programa vuelca la información del listado de paquetes en el fichero escogido.

Caso de uso 4: "abrir captura"

El usuario elige la opción del menú "abrir". Seguidamente visualiza una pantalla de navegación donde se puede especificar el fichero a abrir. Una vez haga click en el botón "abrir", el programa vuelca la información del paquete en una lista de paquetes.

3.3 Diagrama estático de diseño

El diagrama estático de diseño queda de la siguiente manera:

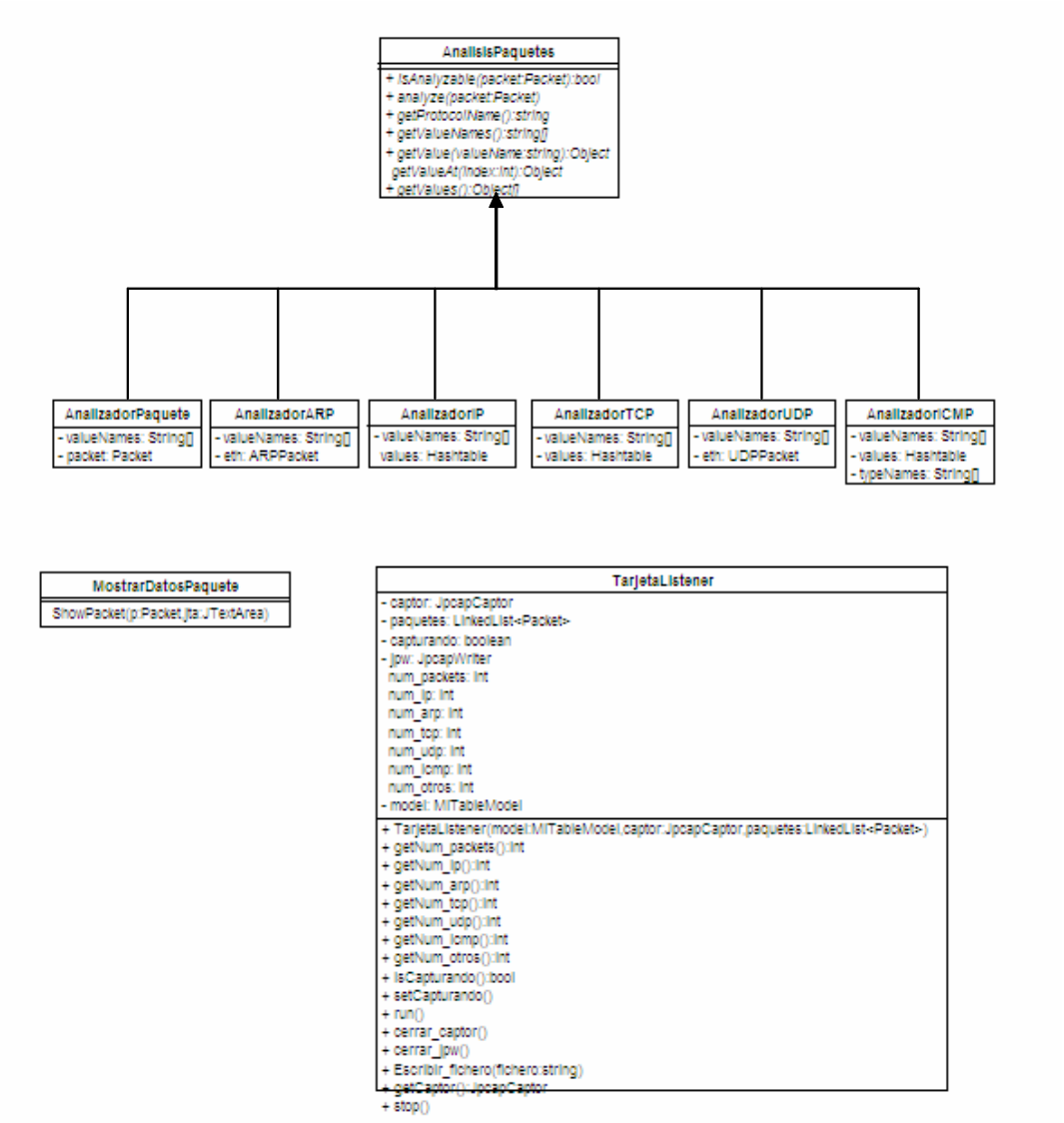


Figura 6.- Diagrama de clases

3.4 Diseño de la interface de usuario

3.4.1 Presentación de los datos

Los datos básicos de los paquetes se presentan en forma de tabla. En cada línea de la tabla le corresponde un array con todos los datos básicos del paquete. Se puede desplazarse por la tabla verticalmente.

El campo de datos del paquete se presenta en forma de área de texto y los campos del paquete a los distintos niveles se presentan en forma de árbol, donde cada nodo representa un nivel.

3.4.2 Implementación de diálogos

Mediante un menú se implementa las distintas opciones de que dispone la aplicación, el menú aparece en la parte superior de la pantalla principal del programa.
También se dispone de una ventana de navegación para abrir y otra para guardar sesiones de capturas, en donde se escoge que sesión se quiere abrir o donde se guarda esa sesión.

CAPÍTULO 4. LA IMPLEMENTACIÓN DEL PROYECTO

4.1 La librería Jpcap

Para la realización de este proyecto, se necesitaba algún tipo de herramienta que facilitara de alguna forma su implementación. Se ha buscado en Internet posibles soluciones y de todas ellas la más idónea ha sido la utilización de Jpcap.

Jpcap, como se comenta posteriormente en el Anexo A, es una librería escrita en Java que nos ofrece las siguientes funcionalidades:

a) Listado de las tarjetas de red disponibles en la máquina (método getDeviceList())
b) Iniciar sesión de captura con la tarjeta de red escogida (método openDevice())
c) Recuperar una sesión en el formato de fichero tcpdump (método openFile())
d) Recuperar cada paquete de la sesión (método getPacket ())
e) Métodos para recuperar la información de cada paquete
f) Salvar en un fichero tipo tcpdump una sesión de captura. (método JpcapWriter() y WriterDumpFile()).

4.2 Las librerías graficas

4.2.1 La librería Swing

La librería Swing de Java proporciona todos los componentes necesarios para la construcción de la interface gráfica de usuario. Los componentes Swing frente a los componentes AWT (llamados componentes pesados) presentan la ventaja de ser independientes de la plataforma, por eso reciben el nombre de componentes ligeros. Los componentes Swing son más numerosos que los AWT (por ejemplo, fichas, paneles con scroll, árboles).

Swing basa sus componentes en la arquitectura *modelo-vista-controlador (MVC).* El modelo se corresponde con el estado del componente; por ejemplo, en una lista el modelo está definido por todos los elementos de la misma; en cambio, en una caja de texto el modelo está definido por el documento de texto. La vista se refiere a cualquier perspectiva de ese modelo; por ejemplo, en una lista la vista es la representación gráfica de la misma, y en una caja de texto, ídem. Y el controlador (manejador de eventos) es el responsable de la actualización del modelo.

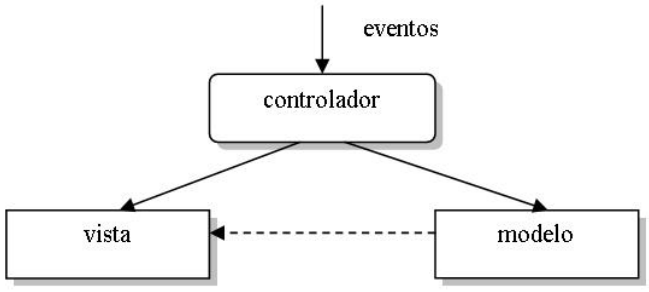


Figura 7.- Arquitectura modelo-vista-controlador

Tanto la vista como el controlador son combinados en un objeto denominado *delegado.* Esto es, el delegado representa gráficamente el modelo (trabajo que hace la vista) y transfiere la entrada del usuario hacia este (trabajo que hace el controlador). De esta forma evitamos conocer aspectos internos muy complejos, como la comunicación entre el controlador y la vista.

Los componentes Swing se pueden anidar unos dentro de otros y permiten crear interfaces más atractivas y con mayor funcionalidad que los componentes AWT. Por estas razones ha sido escogida para la creación de la interface gráfica de usuario de la presente aplicación.

4.2.2 La librería JFreeChart

JFreeChart es una librería de gráficos escrita en Java. Esta diseñada para usar en aplicaciones, applets, servlets y JPS. JFreeChart puede generar gráficas de distintas formas, tarta, barras, líneas, etc.

Gracias a esta librería, la aplicación muestra en el apartado “Estadísticas” de la barra de menú, unas gráficas en forma de tarta de los niveles de Internet y transporte, donde se refleja el peso de cada protocolo en cada una de las capturas realizadas.

4.3 Decisiones tomadas en la implementación del programa

La primera decisión a tomar es el número de paquetes máximo que se quiere capturar en una sesión. En un principio se decide por un número de 1000 paquetes pero posteriormente se amplia a 3000 paquetes, ya que es una cantidad perfectamente asumida por la aplicación. Otro factor a tener en cuenta es el momento en que se aplica el filtro de paquetes, en esta aplicación se realiza antes de comenzar la captura.

A partir que la aplicación empieza a capturar paquetes, se ha optado por que el usuario no pueda interactuar con el programa hasta que esta captura alcance el número máximo de paquetes que se pueden capturar o el usuario paré voluntariamente la captura.
Esto se ha conseguido con la aparición de una pantalla donde se puede observar el número y el tipo de paquetes que captura en tiempo real junto con una barra de progreso. De esta pantalla no se puede salir hasta que se pulse el botón stop, momento en el cual se para la captura y se puede interactuar con los paquetes capturados.

Todos los paquetes capturados se guardan en un listado para ser analizados posteriormente. Este listado de paquetes servirá para el posterior almacenaje en un fichero si fuera el caso.

Otro aspecto que se quiere comentar es la representación gráfica de los mensajes ICMP. Los mensajes ICMP viajan dentro de paquetes IP (al contrario de lo que ocurre con los paquetes ARP), y es un protocolo del nivel Internet. Pero para su representación gráfica se han optado por incluir este protocolo en el nivel de Transporte y no tendría sentido que estuvieran representados en el nivel de Internet ya que todo mensaje ICMP también es un paquete IP.

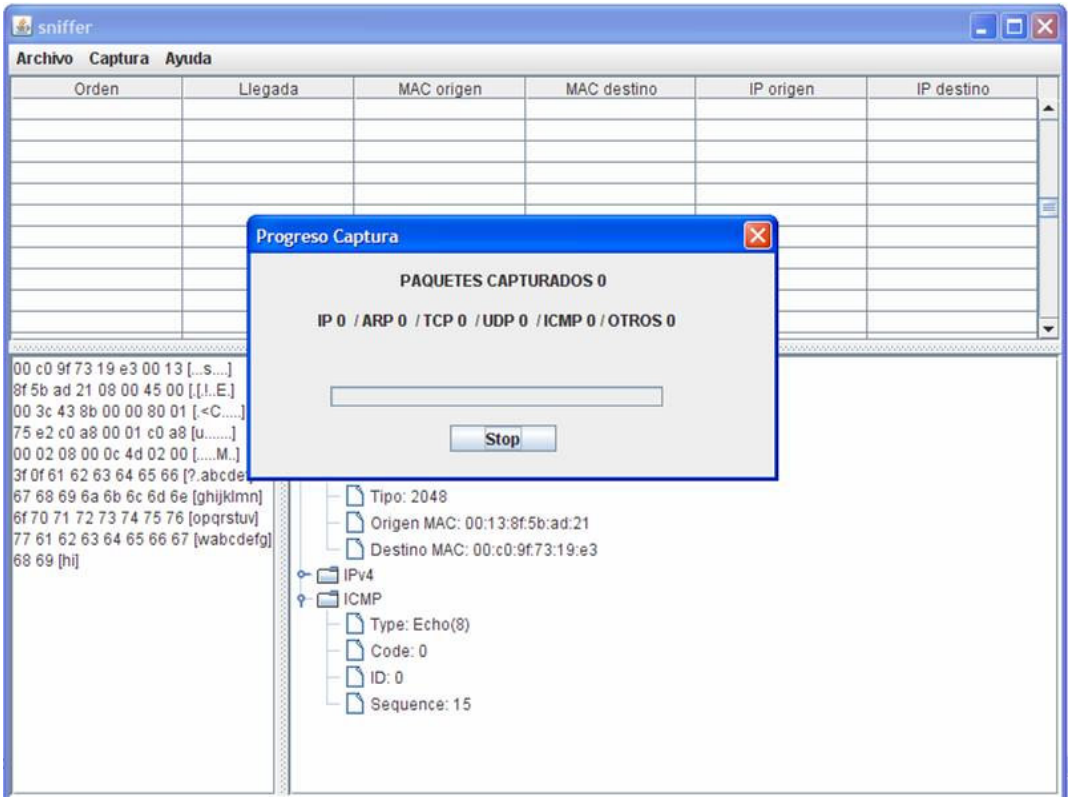


Figura 8.- Momento en el que se esta realizando una captura

4.4 filtrado de paquetes

Se han estudiado diversas maneras de indicar al programa el filtro a aplicar antes de realizar la captura, y de todas ellas se ha optado la opción de escribir un String dentro de una caja de texto que se encuentra en la pantalla de opciones del programa.

Este String debe tener el mismo formato que el utilizado en el programa Tcpdump, en el caso de que este String no coincida con el formato de filtros de Tcpdump, la aplicación muestra un mensaje de error. El trozo de código que pasa el filtro a la captura es el siguiente:

```
if(filterField.getText() != null&& filterField.getText().length() > 0)
{
      jpcap.setFilter(filterField.getText(), true);
      }
```

Donde “filterField.getText ()” se encarga de capturar el String que el usuario ha introducido en la caja de texto pertinente.

4.5 Visualización de los paquetes

Una vez finalizada la captura de los paquetes en tiempo real, o a partir de un fichero, empieza lo que seria la visualización de los paquetes.

En el panel donde se encuentra la tabla, aparecen los datos básicos de los paquetes, que son el número de orden, el tiempo de llegada, MAC origen, MAC destino, IP origen y IP destino.

Debajo de la tabla se encuentra dos paneles más en donde se visualizan los paquetes de forma detallada. Este contenido consiste en los datos del paquete en formato hexadecimal sin entrar en más detalles y los campos de las cabeceras de cada paquete a distintos niveles, que son:

a) Un nivel de información del paquete
b) Un nivel de información del paquete Ethernet
c) Un nivel de información del paquete IP o ARP
d) Un nivel de información del paquete TCP, UDP o ICMP

Para la realización del panel donde se muestra los datos básicos del paquete se utiliza una JTable. Un JTable cuenta con un constructor que crea un modelo predeterminado de tabla a partir de una serie de datos.

Para la realización del panel donde se muestra los datos del paquete se ha utilizado un componente JTextArea, pasando los datos a formato hexadecimal.

Para la realización del panel donde se muestran los campos de las cabeceras se ha utilizado un objeto JTree (árbol). Este componente visualiza una lista jerárquica de elementos, denominados nodos, compuestos cada uno de ellos por una etiqueta y un icono.

4.6 Guardar una sesión de captura en un fichero

A la vez que se realiza la captura de paquetes, la aplicación va guardando los paquetes que recibe en un “LinkedList <Packet>”. El mecanismo para pasar los elementos del LinkedList al fichero es el siguiente:

```
public void Escribir_fichero(String fichero) {
            try {
                  jpw = JpcapWriter.openDumpFile(captor, fichero);
            } catch (IOException e) {
                  e.printStackTrace();
            }
            for (Packet p : paquetes) {
                  jpw.writePacket(p);
            }
            jpw.close();
            captor.close();

      }
```

Este fichero tiene el formato estándar del Tcpdump y se puede abrir desde una aplicación comercial, como por ejemplo Ethereal. La ventana de navegación se ha implementado con un componente “JFileChooser” de la librería Swing.

4.7 Lectura de un fichero de sesión de captura

El resultado de la lectura del fichero aparece de igual manera que cuando realizamos una captura, es decir, en la tabla de la ventana principal de la aplicación se insertan filas a medida que se obtienen cada uno de los paquetes que contiene el fichero.

El mecanismo de apertura de un fichero es el siguiente:

```
try {
JpcapCaptor captor = JpcapCaptor.openFile(path);

      while (true) {
            Packet p = captor.getPacket();
            if (p == null || p == Packet.EOF)
                  break;
            paquetes.add(p);
      }
      captor.close();

} catch (IOException e1) {
      JOptionPane.showMessageDialog(null,
            "No se puede abrir el fichero: " + path);
      return;
}
```

Se observa que paralelamente también se guarda cada paquete en un LinkedList, ya que es necesario recuperarlo para mostrar la cabecera del paquete detalladamente en el árbol y los datos del paquete en el panel destinado a ello.

CAPÍTULO 5. PRUEBAS

5.1 Primera prueba

Escenario: Dentro de una red local, se esta haciendo un ping permanente a la máquina “192.168.1.10”.

Objetivo: Capturar los paquetes ICMP que tengan como IP origen nuestra máquina “192.168.1.116”.

La pantalla de filtro queda de la siguiente manera:

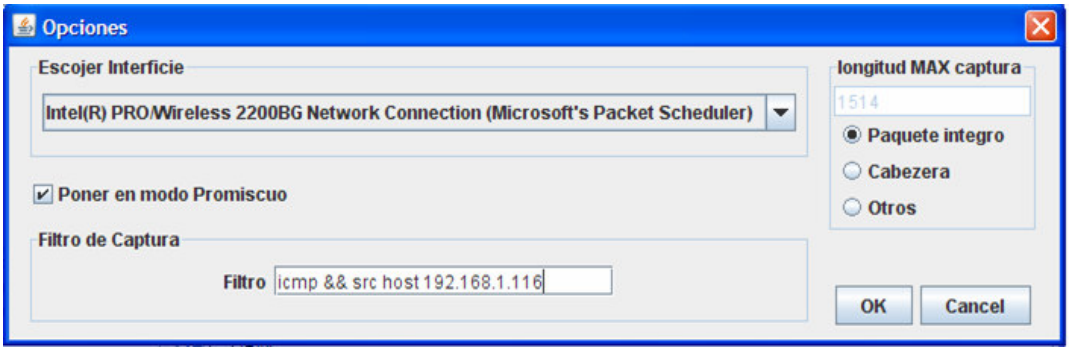


Figura 9.- Pantalla opciones primera prueba

Una vez finalizada la captura el resultado seria el siguiente:

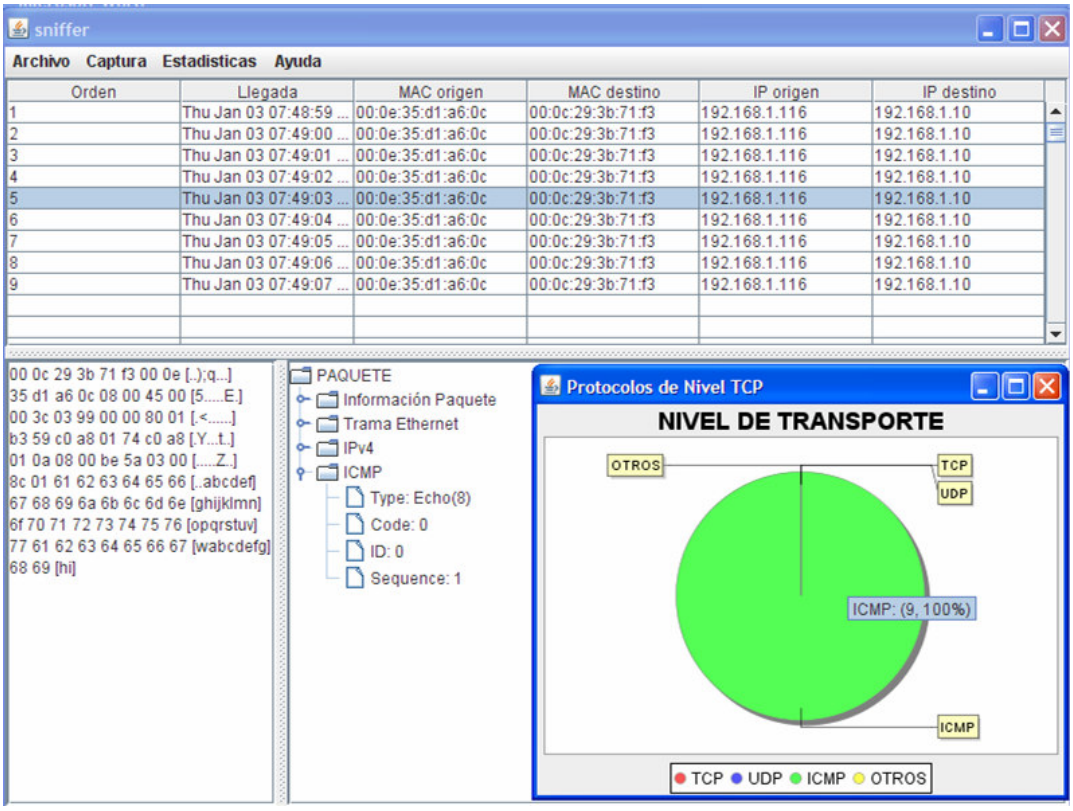


Figura 10.- Pantalla captura primera prueba

Se puede observar como el gráfico muestra que se han capturado 9 paquetes ICMP, representando el 100% de la captura.

5.2 Segunda prueba

Escenario: Dentro de una red local, se esta haciendo un ping permanente a la máquina “192.168.1.10” y a la vez se accede al servidor apache instalado en dicha máquina.

Objetivo: Capturar solo los paquetes TCP y la longitud de captura del paquete no exceda de 100 bytes.

La pantalla de filtro queda de la siguiente manera:

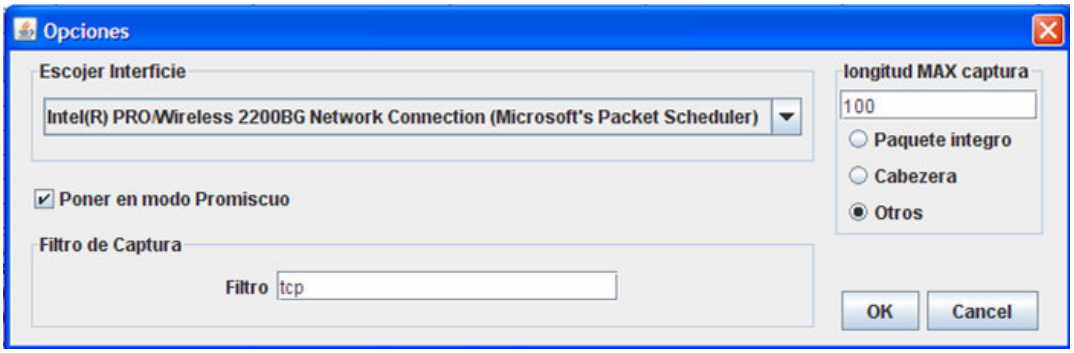


Figura 11.- Pantalla opciones segunda prueba

Una vez finalizada la captura el resultado seria el siguiente:

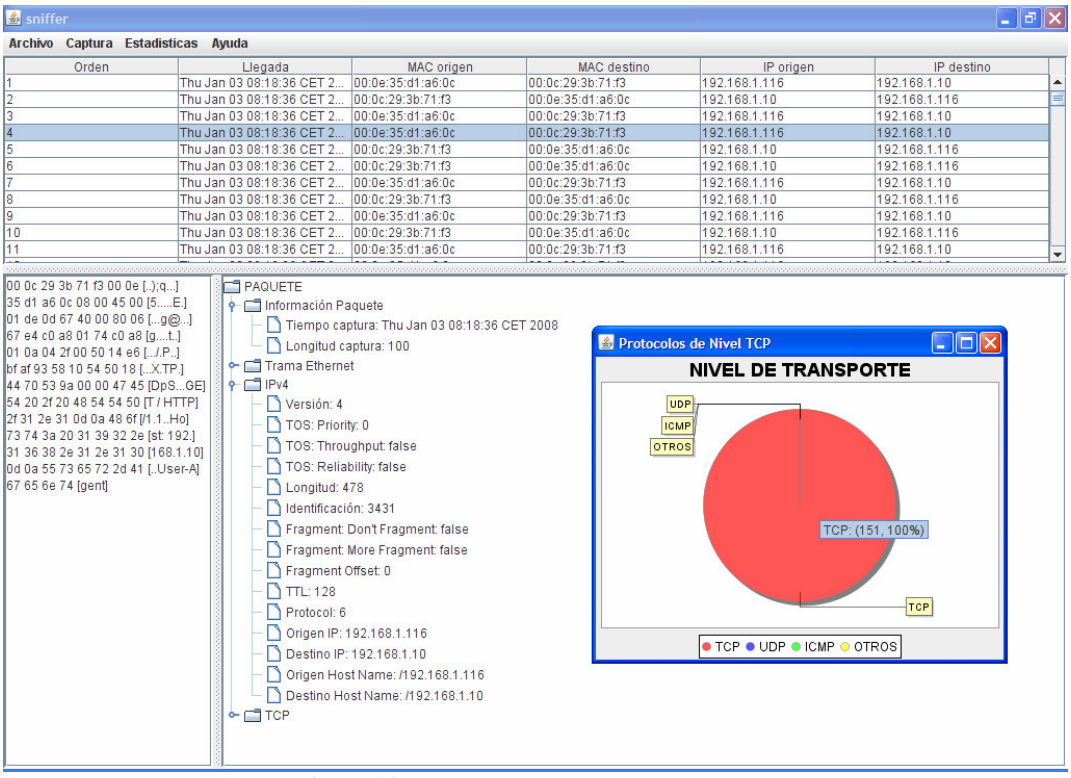


Figura 12.- Pantalla captura segunda prueba

Como se puede ver, la gráfica muestra como todos los paquetes capturados son TCP y en el nodo del árbol “Información Paquete” la longitud de la captura es de 100 bytes, mientras que en el nodo “IPv4” la longitud del paquete es de 478 bytes.

5.3 Tercera prueba

Escenario: se realiza una captura con el sniffer ethereal y se guarda en un fichero que se llama “captura_ethereal”.

Objetivo: Abrir el fichero con nuestro sniffer y poder interactuar con él.

La pantalla del sniffer ethereal que muestra la captura es la siguiente:

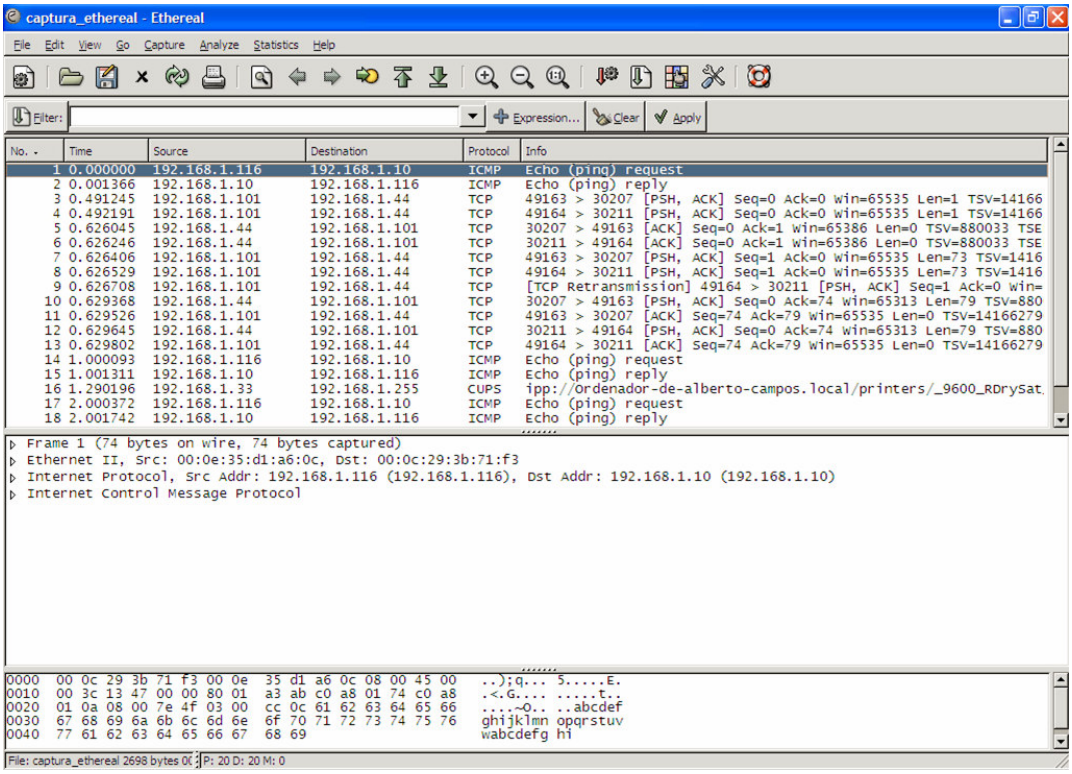


Figura 13.- Pantalla captura ethereal

Una vez abierto el fichero en el sniffer del presente TFC, podemos observar que tanto en la pantalla del ethereal como la del sniffer tenemos seleccionado el primer paquete y que los datos coinciden.

Otra cosa importante a destacar es que hay paquetes que Ethereal puede analizar al completo, mientras que el sniffer los captura pero no los puede analizar al completo, solamente ofrece información del paquete y de la trama Ethernet. Esto quiere decir que el sniffer no puede analizar el protocolo que se encuentra encapsulado en la trama Ethernet, también se puede apreciar en la gráfica a nivel Internet que se muestra en la captura.

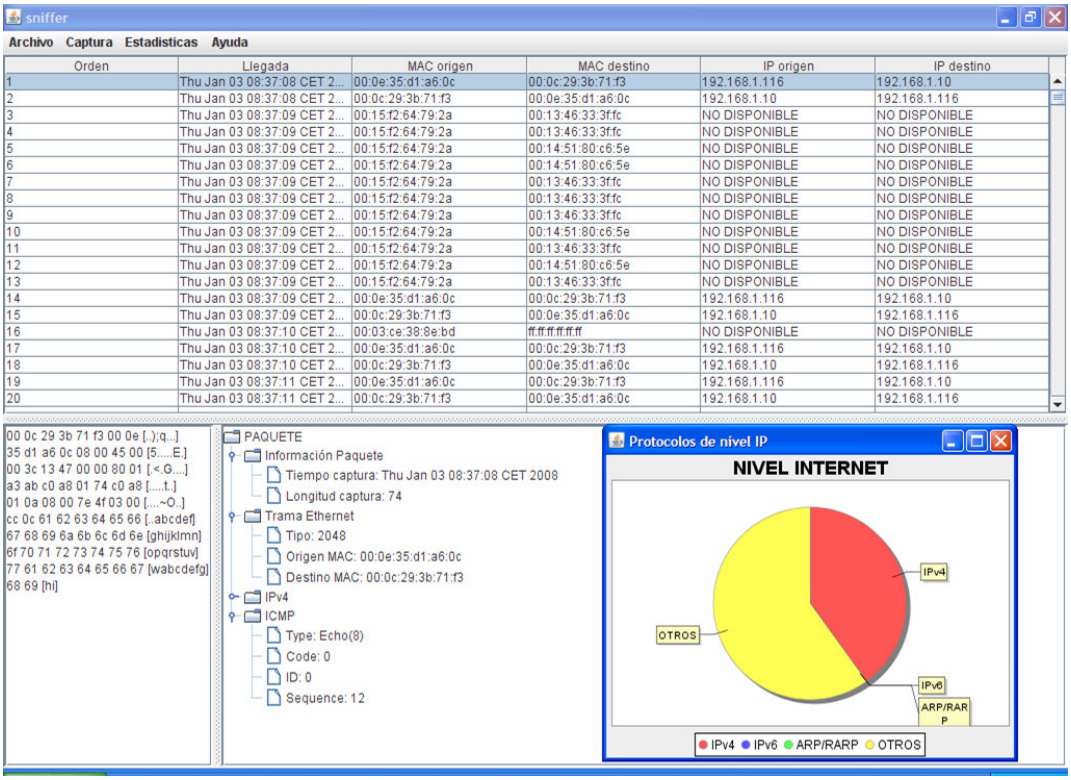


Figura 14.- Pantalla de captura del sniffer

5.4 Cuarta prueba

Escenario: se está navegando con Mozzilla Firefox.

Objetivo: Capturar los paquetes cuyo puerto de destino sea el 80 y hacia la máquina www.uoc.edu.

La pantalla de filtro de captura queda de la siguiente manera:

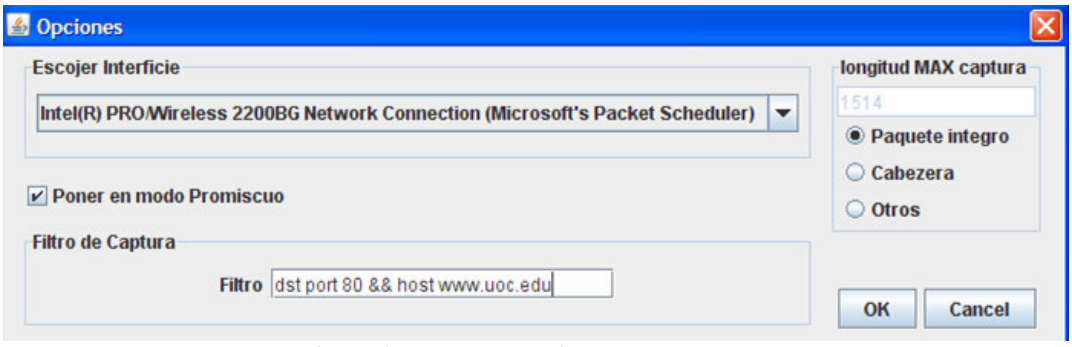


Figura 15.- Pantalla opciones cuarta prueba

La máquina se puede especificar de dos maneras:

- Con la IP de la máquina
- Con el nombre de la máquina

En este caso se ha optado por el nombre de la máquina, ya que se desconoce su ip.

El resultado de la captura es el siguiente:

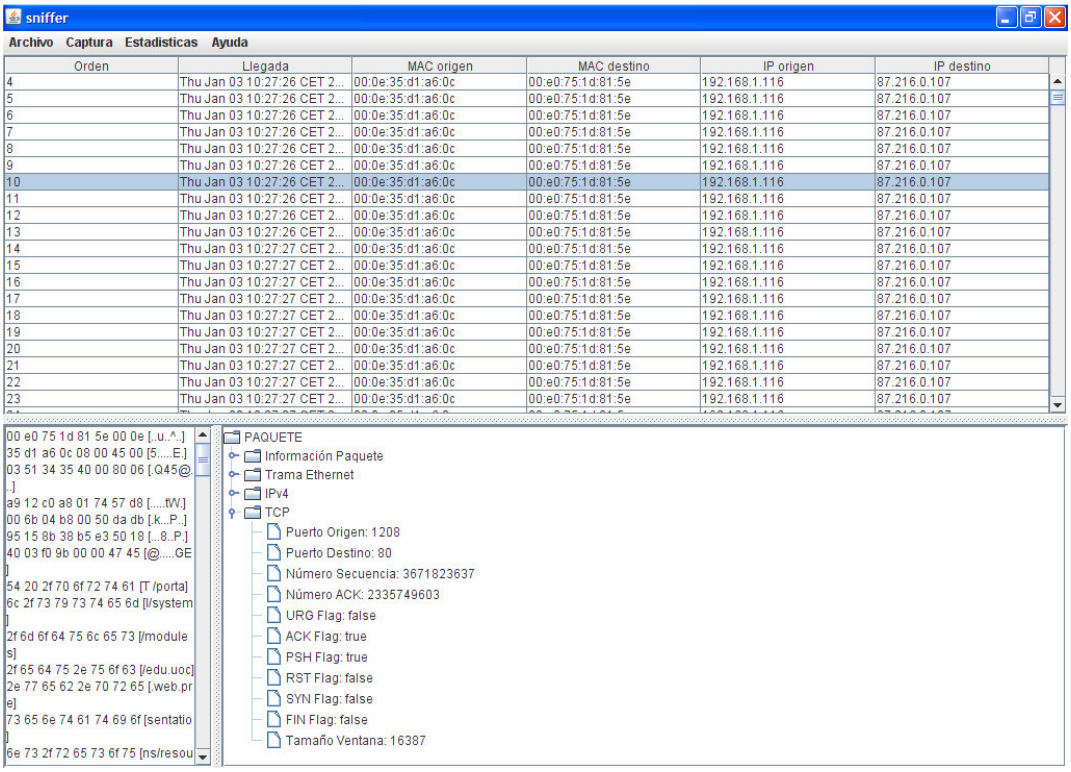


Figura 16.- Pantalla captura cuarta prueba

Se observa en un paquete seleccionado al azar que el puerto destino es el 80 y la ip destino es igual para todos los paquetes, en este caso la ip de la máquina www.uoc.edu es “87.216.0.107”.

Las capturas realizadas para la efectuar las pruebas se han guardado en los siguientes archivos:

1. prueba1
2. prueba2
3. captura_ethereal
4. prueba4

Que se adjuntan en la carpeta “pruebas” como material entregado junto a la memoria y el sniffer.

CAPÍTULO 6. MANUAL DE INSTALACIÓN Y USUARIO

Este programa ha sido testado en un Intel® Pentium® M processor 1.60GHz, 496 MB de RAM, con sistema operativo Microsoft Windows XP Profesional Versión 2002, Service Pack 2.

6.1 Manual de instalación

<MICROSOFT WINDOWS>

Se necesita tener instalado:

- Librería winPcap (http://www.winpcap.org)
- Librería Jpcap (http://netresearch.ics.uci.edu/kfujii/jpcap/doc/index.html)
- JRE 6 (http://java.sun.com)

<LINUX> (Ubuntu, GNU/Debian)

Bajar e Instalar Jpcap Debian package.
Para algunas distribuciones (Ubuntu), JDK6 también se instala automáticamente.

PARA EJECUTAR EL PROGRAMA:

La aplicación se distribuye en un archivo comprimido WinRAR llamado "sniffer". Descomprimir el archivo en un directorio creado para tal fin y posteriormente existen dos opciones para ejecutarlo:

- desde la línea de comando, ir a la carpeta "dist" y teclear lo siguiente:
  java –jar sniffer.jar

- abrir la carpeta "dist" y hacer doble clic en el fichero sniffer.jar

6.2 Manual de usuario

Una vez ejecutado el programa se visualiza una pantalla principal en el cual se puede interactuar con un menú contextual que dispone de las siguientes opciones:

- Archivo
  - Abrir
  - Guardar
  - Salir
- Captura
  - Start
- Estadísticas
  - Información general
  - Nivel Internet
  - Nivel de transporte
- Ayuda
  - About

6.2.1 Iniciar una nueva captura

Activar la opción del menú “Captura > Start”.
En este paso entran en juego diferentes ventanas; después de seleccionar Start del menú captura aparece una ventana de Opciones:

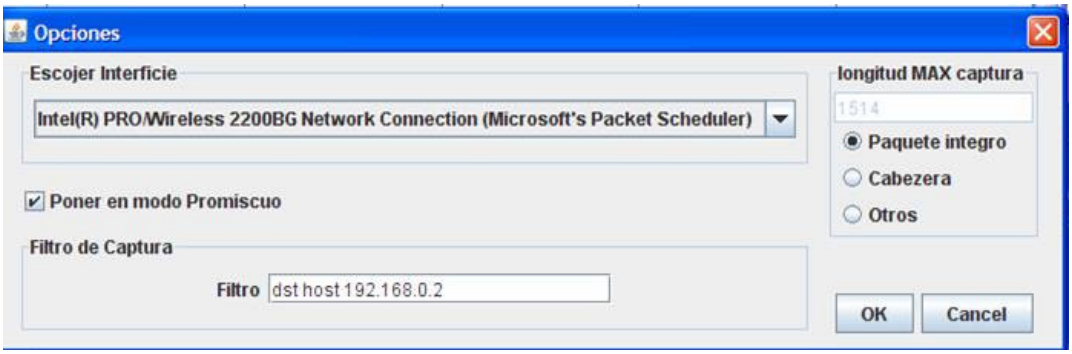


Figura 17.- Pantalla de opciones

Estas opciones consisten en escoger la tarjeta de red desde la cual se realiza la captura, el filtro que quiere aplicar, la longitud máxima del paquete a capturar y el modo de captura (promiscuo o no promiscuo).

La longitud máxima del paquete a capturar se puede escoger entre tres opciones:

- Paquete integro
- Solo la cabecera
- La longitud que se desee (entre 68 y 1514 bytes)

El filtro de captura se introduce a través de un String, cuyo formato es el mismo que utiliza tcpdump, por lo tanto, el usuario tiene que tener nociones sobre este tema. En el “anexo c” se explica la sintaxis de estos filtros. En el caso de que el filtro introducido no sea correcto se muestra una pantalla de error.

Después de confirmar la pantalla Opciones, aparece una ventana donde se va indicando la cantidad y la clase de los paquetes capturados.

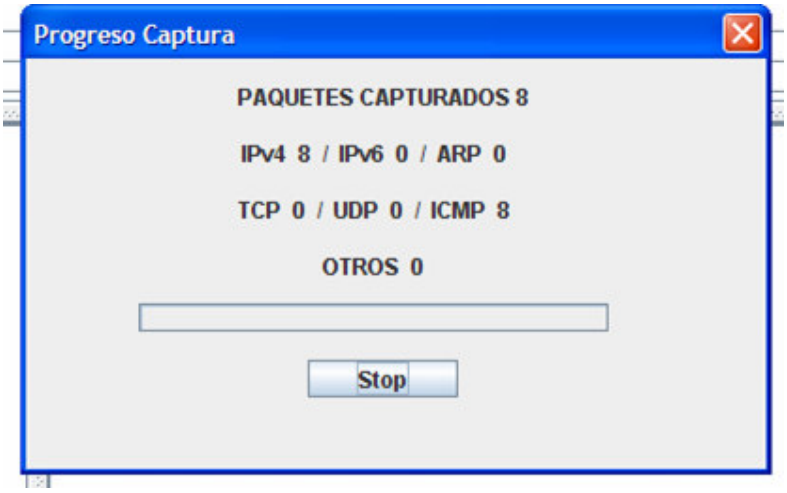


Figura 18.- Pantalla de progreso.

Es en esta pantalla donde se puede parar la captura cuando se desee (botón Stop) o esperar a que se alcance al número máximo de paquetes permitido.

Después de esto, ya se puede interactuar con la tabla que contiene los paquetes capturados y visualizar con detalle los datos de cada paquete.

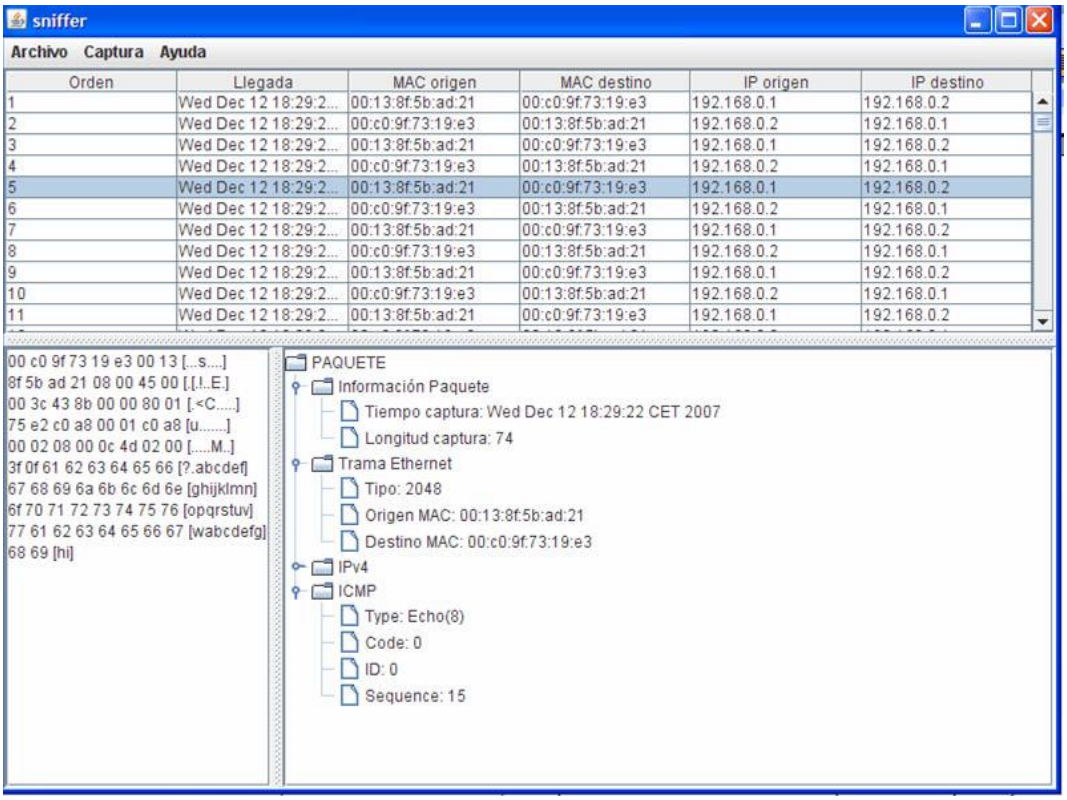


Figura 19.- Pantalla Principal

Para ver con detalle un paquete se tiene que clickar sobre una fila de la tabla que contenga un paquete. El programa captura la fila en la cual se ha clickado y realizará la tarea de mostrar el contenido detallado del paquete.

En el panel inferior de la izquierda se muestran los datos del paquete que hemos seleccionado y en el panel inferior de la derecha se muestran en forma de árbol los datos de las cabeceras del paquete desglosado en los distintos niveles que encapsula la información.

6.2.2 Guardar una captura

Activar la opción del menú “Archivo > Guardar”.
Aparece una ventana de dialogo donde se puede escoger la ubicación y el nombre del fichero a guardar.

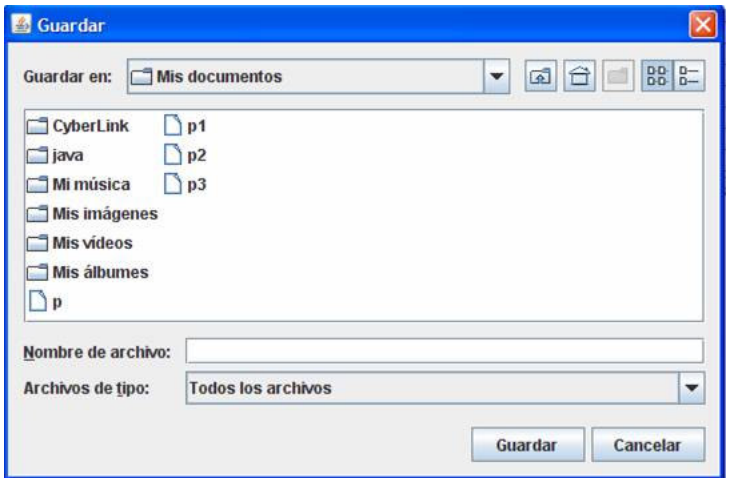


Figura 20.- Guardar fichero

6.2.3 Abrir un fichero

Activa la opción del menú “Archivo > Abrir”.
Aparece una ventana de dialogo igual a la de la figura 12, donde se puede escoger el fichero que queremos abrir, si este no puede ser abierto por la aplicación se visualiza un panel donde se informa de dicho suceso.

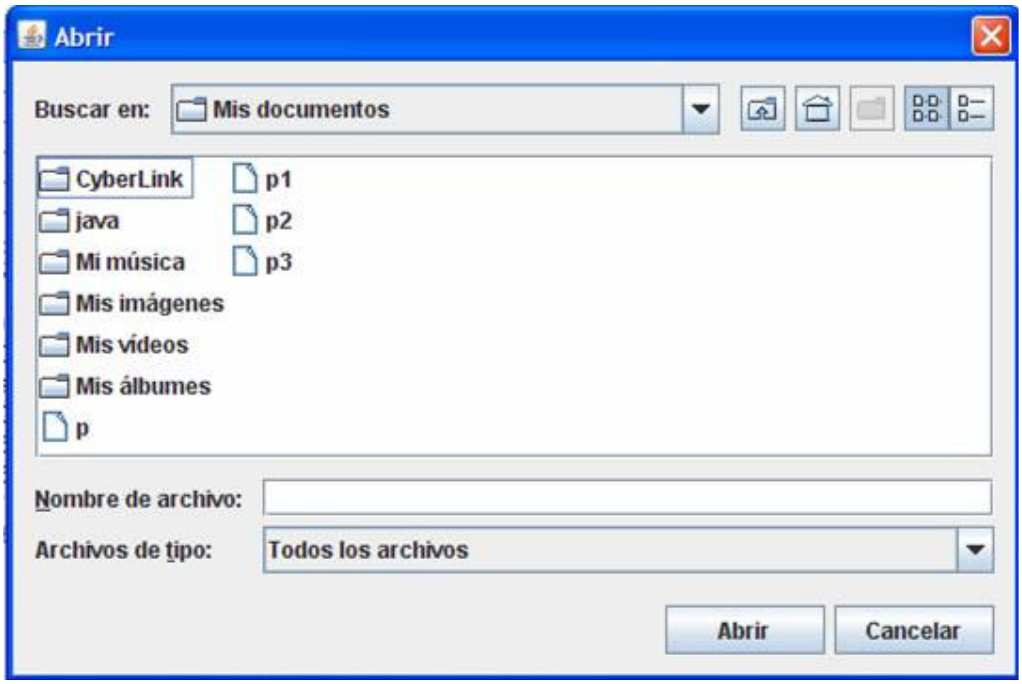


Figura 21.- Abrir fichero

6.2.4 Salir del programa

Hay dos maneras de salir de la aplicación:

- Archivo > Salir
- Clickando sobre el icono de la cruz situado en la parte superior derecha de la ventana principal del programa

6.2.5 Mostrar estadísticas

En el menú “Estadísticas” hay tres opciones como se ha indicado al principio del apartado. La opción “Información general” nos muestra la cantidad de paquetes, la suma total de la longitud de los paquetes y la longitud media de los paquetes capturados.

La opción “Nivel Internet” muestra una gráfica en forma de tarta con la cantidad y el porcentaje de los protocolos en este nivel.

La opción “Nivel de transporte” muestra igualmente una gráfica con la cantidad y el porcentaje de los protocolos en este nivel. En esta gráfica también se incluyen los mensajes ICMP, que aunque no pertenecen a este nivel, se ha optado por trasladarlos al nivel de transporte tal y como se explica en el apartado “4.3 Decisiones tomadas para la implementación del programa”.

6.2.6 Ayuda

Ayuda > About

Esta opción simplemente muestra un panel informativo indicando el nombre del estudiante y el tema donde se enmarca el presente TFC (Xarxes).

CONCLUSIONES

El sniffer desarrollado cumple todos los requisitos que en un principio se habían planteado como objetivo de este TFC y se han aplicado los conocimientos adquiridos a lo largo de la carrera.

Principalmente el desarrollo de una aplicación como esta lleva consigo un número de horas de trabajo importante y la adquisición de nuevos conocimientos para solucionar problemas que se presentan a lo largo del proyecto.

Sin duda una de las posibles mejoras del sniffer es la de poder analizar más cantidad de protocolos. Otro punto que se puede mejorar es el tema estadístico, si se compara con un analizador como Ethereal se observa que en este sentido queda mucho por hacer.

También se podría añadir la opción de impresión, ya sea del contenido de uno o de diferentes paquetes seleccionados o bien la impresión del listado de todos los paquetes con sus datos básicos de una sesión de captura. Hay que tener en cuenta que este Trabajo Final de Carrera se ha realizado solo en un cuatrimestre y todas estas mejoras serian posibles con más tiempo disponible.

En este proyecto ha habido tres puntos problemáticos a la hora de desarrollar el proyecto:

1. Creación de la interface gráfica de usuario.
2. Realizar la captura en tiempo real.
3. Mostrar algún tipo de gráfico que muestre los resultados de la captura.

El primer punto más que un problema, era un cierto desconocimiento sobre como desarrollar una interface gráfica de usuario, ya que no se había tocado este tema durante el transcurso de la carrera. Para ello se realizó un estudio de la librería Swing de Java que conllevó a la ampliación en una semana del paso 3 (“implementación del sniffer”) en la planificación del proyecto.

En el segundo punto, se necesitaba poder realizar la captura sin que la respuesta de nuestra Interfaz de Usuario (UI) se viera comprometida. Para resolver esto, se debe utilizar hilos (threads en inglés), estos hilos nos permiten realizar una tarea mientras hacemos otras, lo que significa que nuestra UI no se bloqueará.
Se opta por utilizar la interfaz ExecutorService para gestionar threads. Esta interfaz pertenece al paquete java.util.concurrent y esta disponible a partir de la versión jdk 1.5 de Java.

Para el tercer punto, se quería reflejar los resultados de la captura de forma gráfica, no sé tenia muy claro la forma de hacerlo pero después de buscar soluciones para ello, se optó por la librería JFreeChart, gracia a ella se puede visualizar una gráfica en forma de tarta para los diferentes niveles y simultáneamente ver datos de captura manteniendo el ratón sobre la gráfica.

GLOSARIO

Protocolo: descripción formal de formatos de mensajes y reglas que dos o más máquinas deben seguir para intercambiar mensajes.

Paquete: en términos generales, cualquier bloque pequeño de datos enviado a través de una red de conmutación de paquetes.

ARP: protocolo TCP/IP utilizado para asignar una dirección IP de alto nivel a una dirección de hardware físico de bajo nivel.

Ethernet: Es el protocolo de red de nivel 2 más utilizado en redes LAN (Local Area Networks). Existen otros protocolos de red de niveles superiores que pueden ser transportados por el protocolo Ethernet (como ser TCP/IP).

IP: unidad básica de información que pasa a través de una red de redes TCP/IP. Contiene las direcciones de fuente y destino junto así como los datos.

TCP: protocolo de nivel de transporte TCP/IP estándar que proporciona el servicio de flujo confiable full duplex y del cual dependen muchas aplicaciones.

UDP: protocolo estándar TCP/IP que permite a un programa de aplicación en una máquina enviar un datagrama hacia el programa de aplicación en otra máquina.

ICMP: parte integral del protocolo de Internet (IP) que resuelve errores y controla los mensajes.

Package: en java, un paquete (package) es una colección de clases relacionadas, según los criterios que desee el programador. En general, las clases se agruparán en paquetes dependiendo de su cometido específico, es decir, de la función para la cual han sido redactadas.

Open Source: código abierto (del inglés open source) es el término con el que se conoce al software distribuido y desarrollado libremente. Fue utilizado por primera vez en 1998 por algunos usuarios de la comunidad del software libre, tratando de usarlo como reemplazo al ambiguo nombre original en inglés del software libre (free software).

BIBLIOGRAFÍA

Jpcap “a Java library for capturing and sending network packets”
http://netresearch.ics.uci.edu/kfujii/jpcap/doc/index.html

Ethereal “The world’s most popular network protocol analyze”
http://www.etehreal.com

Tcpdump/libpcap
http://www.tcpdump.org

JFreeChart
http://www.jfree.org/jfreechart

WinPcap: The Windows Packet Capture Library
http://www.winpcap.org

JavaHispano
http://www.javahispano.org

Java en castellano
http://www.programacion.com/java

Programación de hilos en Java
http://roberto.stinkybug.cl/blog/2006/07/22/programacion-de-hilos-en-java

Ngrep – network grep
http://ngrep.sourceforge.net

Ettercap
http://ettercap.sourceforge.net

Kismet
http://www.kismetwireless.net

Francisco Javier Ceballos (2006). *Java 2 Interfaces gráficas y aplicaciones para Internet, 2ª Edición*. Madrid: RA-MA.

Pedro Manuel Cuenca Jiménez (2000). *Programación en Java*. Ediciones Anaya Multimedia.

Muñoz Caro, C.; Niño Ramos, A; Vizcaíno Barceló, A. (2002) *Introducción a la programación con orientación a objetos.* Pearson Educación, S.A. Madrid.

ANEXOS

Anexo A. La librería JPCAP

Jpcap es una librería *open source* que permite capturar y enviar paquetes de red desde aplicaciones Java. En concreto proporciona facilidades para:

- Capturar paquetes en tiempo real.
- Salvar los paquetes capturados en un fichero, y leer los paquetes capturados desde un fichero.
- Automáticamente identifica los tipos de paquetes y genera los correspondientes objetos Java (para Ethernet, IPv4, IPv6, ARP/RARP, TCP, UDP y ICMPv4 paquetes).
- Filtrado de paquetes de acuerdo con la reglas especificadas por el usuario en la aplicación.
- Envío de paquetes a la red.

Jpcap esta basada en *libpcap/winpcap*, y esta implementada en lenguaje C y Java. Esta librería se compone de dos *packages*:

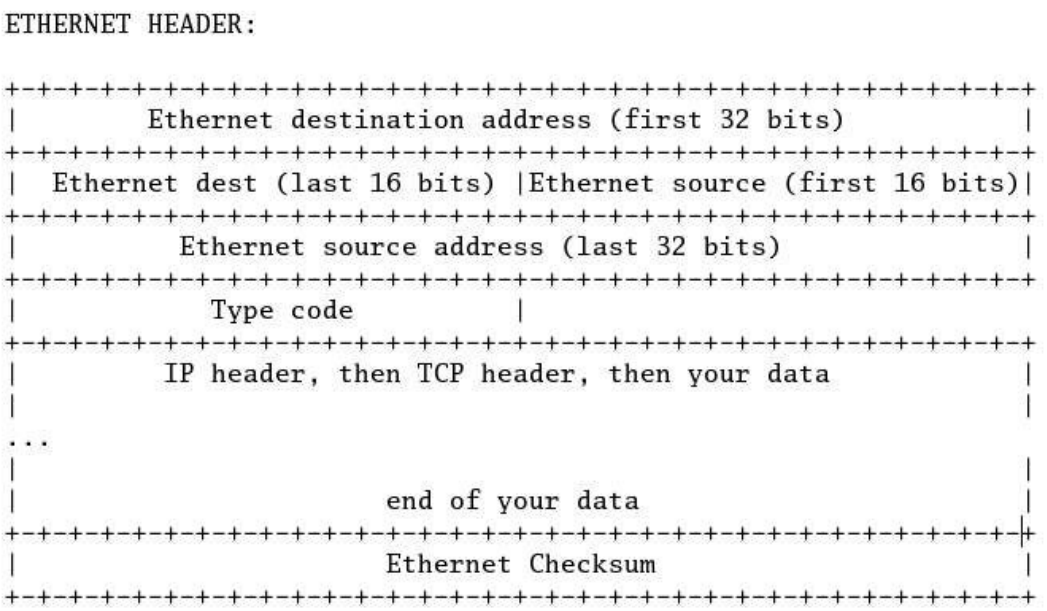
1. Jpcap: proporciona clases e interfaces para capturar paquetes desde una interface de red, enviar paquetes a una interface de red, leer paquetes desde un fichero y escribir paquetes en un fichero local.
2. Jpcap.packet: proporciona clases que representan a varios paquetes.

Muestra de un programa, que se sirve de esta librería, el cual permite escoger la tarjeta de red para realizar la posterior captura de paquetes:

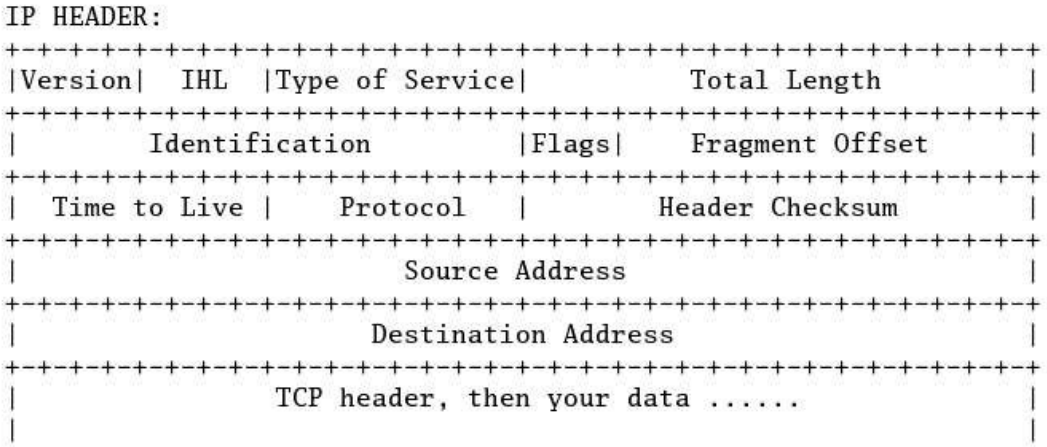
```
import jpcap.*;
class Tcpdump implements JpcapHandler{

      public void handlePacket(Packet packet){
            System.out.println(packet);
      }

      public static void main(String[] args) throws
      java.io.IOException{
            Jpcap jpcap=Jpcap.openDevice(args[0],1000,true,20);
            jpcap.processPacket(-1,new Tcpdump());
      }
}
```

Anexo B. Formato de los paquetes estudiados

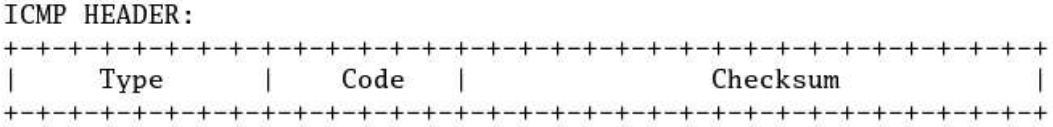
Formato de la trama Ethernet:



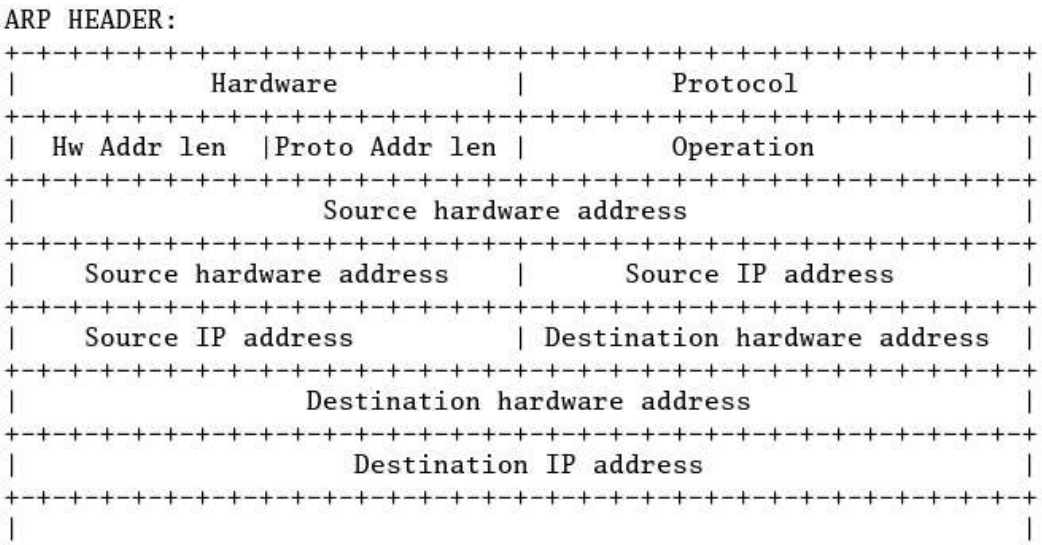
Formato del paquete IP:



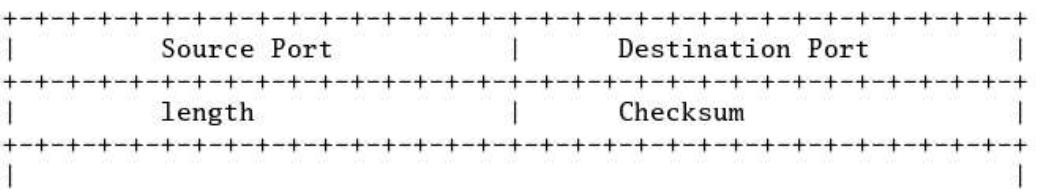
Formato del paquete ICMP



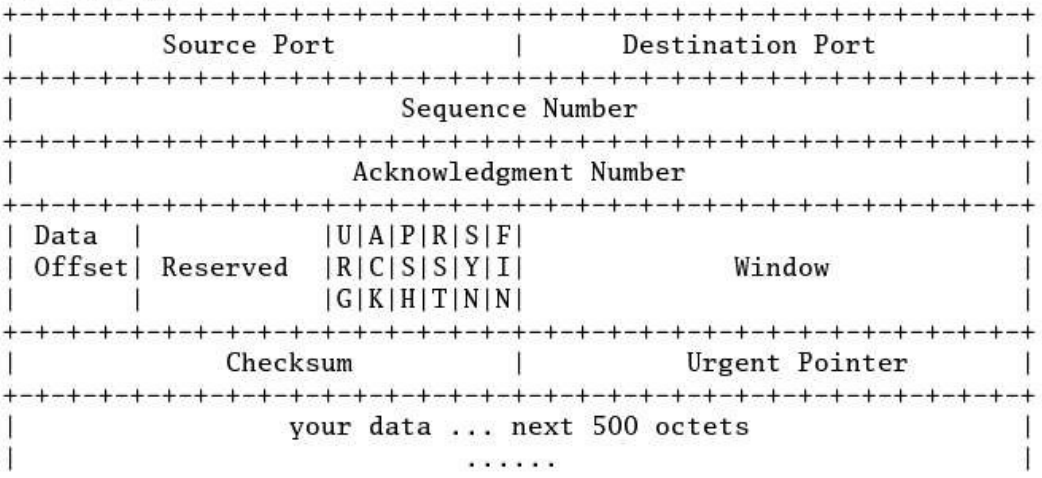
Formato del paquete ARP



Formato del datagrama UDP



Formato del segmento TCP



Anexo C. Filtros de captura

Los filtros de captura que se utilizan son los mismos que se utilizan con el programa tcpdump y están extensamente explicados en la página de manual de tcpdump (man tcpdump). Aquí solamente se presentará un resumen de la sintaxis de los filtros.

Filtrado por host

Sintaxis	Descripción
host host	host puede ser la IP del host como el nombre del host
src host host	Captura todos los paquetes donde host es el origen
dst host host	Captura todos los paquetes donde host es el destino

Filtrado por puerto

Sintaxis	Descripción
port puerto	Captura todos los paquetes donde puerto es tanto el origen como el destino
src port puerto	Captura todos los paquetes donde puerto es el puerto origen
dst port puerto	Captura todos los paquetes donde puerto es el puerto destino

Filtrado por red

Sintaxis	Descripción
net red	Captura todos los paquetes donde la red es tanto el origen como el destino
src net red	Captura todos los paquetes donde la red es el origen
dst net red	Captura todos los paquetes donde la red es el destino

Filtrado por protocolo

Ethernet

Sintaxis	Descripción
ether proto \[primitiva]	
Ejemplos:	
ether proto \ip o solo **ip**	Captura todos los paquetes ip
ether proto \arp o solo **arp**	Captura todos los paquetes arp
ether proto \rarp o solo **rarp**	Captura todos los paquetes rarp

IP

Sintaxis	**Descripción**
ip proto \[primitiva]	
Ejemplos:	
ip proto **tcp** o solo **tcp**	Captura todos los segmentos TCP
ip proto **udp** o solo **udp**	Captura todos los paquetes UDP
ip proto **icmp** o solo **icmp**	Captura todos los paquetes ICMP

Combinando expresiones primitivas
Negación: **!** o **not**
Concatenación: **&&** o **and**
Disyunción:|| o **or**

Ejemplos	
host 10.10.10.10 **&& !net** 192.168	Captura todos los paquetes hacia o desde el host 10.10.10.10 y que no son hacia o desde la red 192.168.0.0
host 10.10.10.10 **&& port** 80	Captura todos los paquetes hacia o desde el host 10.10.10.10 y cuyo origen o destino es el puerto 80