Herramienta de análisis de protocolos de una red.

Alumno: Vicente Almenara Esteve
Grado de Ingeniería Informática

Consultor: María Isabel March Hermo

Data Entrega: 06/2019

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-CompartirIgual 3.0 España de Creative Commons

FICHA DEL TRABAJO FINAL

Título del trabajo:	Herramienta de análisis de protocolos de una red.
Nombre del autor:	Vicente Almenara Esteve
Nombre del consultor:	María Isabel March Hermo
Fecha de entrega (mm/aaaa):	06/2019
Área del Trabajo Final:	Herramienta de análisis de rendimiento de una red
Titulación:	*Grado de Ingeniería Informática*

Resumen del Trabajo (máximo 250 palabras):

Todo administrador de redes se ha encontrado alguna vez ante una pérdida del rendimiento de la red que gestiona. En estos casos, por falta de tiempo o por ignorancia de la existencia de herramientas apropiadas, no es fácil saber las causas por las que está perdida ha ocurrido. La mayoría de veces, las causas de estos problemas se deben a una mala configuración de la red, ataques de terceros, etc.

En todo caso, conocer el principio del problema es el primer paso para poder tomar las medidas necesarias y lograr una correcta protección. Los analizadores de tráfico pueden resultar de gran utilidad para detectar problemas y analizar el tráfico identificando las amenazas de red y así limitar su impacto en la organización.

Este proyecto tiene como finalidad la creación de una herramienta que permitirá capturar y explorar todo el tráfico que se ejecuta por el dispositivo de red seleccionado.

El usuario podrá identificar, analizar y resolver problemas relacionados con los protocolos, redes y la seguridad. Para ello le permitirá filtrar y guardar los paquetes analizados.

A través de este proyecto se pondrá en uso el conocimiento adquirido durante el estudio del grado de ingeniería informática. Se han puesto en práctica conocimientos del área de redes, desarrollo de software, así como, otras áreas como son la gestión de proyectos.

Abstract (in English, 250 words or less):

Every network administrator has ever suffered a loose of performance in the network he is managing. It is then when due to a lack of time or unknowledge, when it is extremely hard to know the cause of this loose. Most of the times the cause of these problems is due to a wrong configuration of the network, outdoor attacks, etc ...

Anyway, to know the beginning of the problem is the first step to take the necesary actions and to achieve the appropriate protection. Data traffic analyzers can be very useful to detect problems. Analysing the data traffic to identify the threats allows us to fence their impact in our organisation.

The aim of this project is to create a tool, that will allow to capture and explore all the traffic that is run by the chosen network device.

User will be able to indentify, analyze and solve problems related with protocols, networks and security. In order to achieve it, the tool will allow to filter and keep the packs already analized.

Through this project we will use the acquired knowledge during this degree of computer engineering. We have put in practice our knowledge and skills in nets, software development, and other fields such as projects management.

Palabras clave (entre 4 y 8):

Sniffer, protocolo, lenguaje C#, paquete, seguridad, captura.

Índice

1. Introducción 1
1.1 Contexto y justificación del Trabajo 1
1.2 Objetivos del Trabajo 2
1.3 Enfoque y método seguido 2
1.4 Planificación del Trabajo 3
1.5 Breve sumario de productos obtenidos 6
1.6 Breve descripción de los otros capítulos de la memoria 6
2. Investigación preliminar 7
2.1 Modelo OSI 7
2.2 Modelo TCP/IP 8
2.3 Protocolos 9
2.4 Encapsulamiento de paquetes 12
2.5 Analizadores de paquetes en el mercado 13
2.5.1 Wireshark 13
2.5.2 Microsoft Message Analyzer 16
3. Análisis y Diseño 18
3.1 Diagrama de clases 18
3.2 Casos de uso 23
4. Implementación 26
4.1 Capturar los paquetes 26
4.2 Modo promiscuo 30
4.3 Ejecutar con permisos de administrador 31
4.4 Filtrado 33
4.5 Gráfica 35
4.6 Guardar 36
5. Manual de usuario 38
6. Caso Práctico 42
7. Conclusiones 46
8. Glosario 47
9. Bibliografía 48
10. Anexos 49
10.1 Primeros pasos En Microsoft Visual Studio 49
10.2 Añadir un paquete NuGet 50
10.3 Crear instalador de la aplicación 51

Tabla de ilustraciones

Ilustración 1. Modelo OSI vs TCP/IP 7
Ilustración 2. Modelo TCP/IP 8
Ilustración 3. Transferencia de un paquete a través de la pila TCP/IP 12
Ilustración 4. Captura 1 - Wireshark 13
Ilustración 5. Captura 2 - Wireshark 14
Ilustración 6. Captura 3 - Wireshark 14
Ilustración 7. Captura 4 - Wireshark 15
Ilustración 8. Captura 5 - Wireshark 15
Ilustración 9. Captura 1 - Microsoft Message Analyzer 16
Ilustración 10. Captura 2 - Microsoft Message Analyzer 17
Ilustración 11. Captura 1 - Captura de paquetes 27
Ilustración 12. Captura 2 - Captura de paquetes 28
Ilustración 13. Captura 3 - Captura de paquetes 29
Ilustración 14. Captura 4 - Captura de paquetes 29
Ilustración 15. Captura - Modo Promiscuo 30
Ilustración 16. Captura 1 - Ejecutar como administrador 31
Ilustración 17. Captura 2 - Ejecutar como administrador 31
Ilustración 18. Captura 3 - Ejecutar como administrador 32
Ilustración 19. Captura 1 - Filtrado 33
Ilustración 20. Captura 2 - Filtrado 33
Ilustración 21. Captura 3 - Filtrado 34
Ilustración 22. Captura 1 - Gráfica 35
Ilustración 23. Captura 2 - Gráfica 36
Ilustración 24. Captura 1 - Guardar 37
Ilustración 25. Captura 2 - Guardar 37
Ilustración 26. Captura 1 - Manual de usuario 38
Ilustración 27. Captura 2 - Manual de usuario 39
Ilustración 28. Captura 3 - Manual de usuario 39
Ilustración 29. Captura 4 - Manual de usuario 40
Ilustración 30. Captura 5 - Manual de usuario 40
Ilustración 31. Captura 6 - Manual de usuario 40
Ilustración 32. Captura 7 - Manual de usuario 41
Ilustración 33. Captura 8 - Manual de usuario 41
Ilustración 34. Captura 9 - Manual de usuario 41
Ilustración 35. Día 15/05 de 08:30 a 10:00 42
Ilustración 36. Día 16/05 de 09:17 a 10:36 42
Ilustración 37. Día 20/05 de 08:27 a 09:47 43
Ilustración 38. Día 22/05 de 11:26 a 13:00 43
Ilustración 39. Día 15/05 Filtro por protocolo IGMP 44
Ilustración 40. Detección problema de seguridad correo 44
Ilustración 41. Detección problema seguridad en archivo guardado 45
Ilustración 42. Configuración seguridad Microsoft Outlook 45
Ilustración 43. Captura 1 - Instalación Visual Studio 49
Ilustración 44. Captura 2 - Instalación Visual Studio 49
Ilustración 45. Captura 1 - Añadir NuGet 50
Ilustración 46. Captura 2 - Añadir NuGet 50
Ilustración 47. Captura 1 - Instalador aplicación 51
Ilustración 48. Captura 2 - Instalador aplicación 51
Ilustración 49. Captura 3 - Instalador aplicación 51
Ilustración 50. Captura 4 - Instalador aplicación 51
Ilustración 51. Captura 5 - Instalador aplicación 52
Ilustración 52. Captura 6 - Instalador aplicación 52
Ilustración 53. Captura 7 - Instalador aplicación 53

1. Introducción

1.1 Contexto y justificación del Trabajo

Hoy en día todas las empresas necesitan tener sus ordenadores conectados y compartir datos para optimizar el proceso de producción. En consecuencia, si la red deja de funcionar y no llegan a transmitirse los datos necesarios, la compañía deja de prestar servicio a los clientes durante el tiempo que dure la caída, con los graves problemas e inconvenientes que esto puede ocasionar.

Además, la información que se transmite a través de las redes circula en forma de paquetes con información. Estos paquetes contienen la información necesaria para que cuando el destinatario los reciba, sea capaz de ordenar y usar la información.

La monitorización de red es una de las responsabilidades más importantes que los profesionales del departamento de tecnología de información tienen dentro de una empresa. No en vano, se trata de gestionar el apartado tecnológico de la compañía ya que, a través de un óptimo monitoreo, se pueden potenciar los resultados obtenidos y aportar una serie de ventajas y obtener una idea muy completa del estado en el que se encuentra la red de la compañía.

Para este proceso, nada mejor que hacer uso de un software de monitoreo de red, una herramienta que nos permitirá detectar, diagnosticar problemas de desempeño de la red mediante su análisis.

El objetivo principal de un software de monitoreo de red, entre otros, es especialmente, la reducción de errores en la red de la empresa. Comprobar, antes que ocurra cualquier problema, la presencia de algún elemento que pueda estar ocasionando una bajada en el rendimiento de la red de la empresa.

En este trabajo se pretende conseguir una aplicación que analice el rendimiento de la red del Hospital General Universitario de Valencia, capturando tráfico de la red y analizando los resultados obtenidos.

La necesidad a cubrir en este trabajo es la prevención del buen funcionamiento de la red del hospital, así como localizar y estudiar posibles fallos de seguridad que existan en dicha red, para poder tomar las medidas necesarias para poderlas solucionar.

1.2 Objetivos del Trabajo

El objetivo principal, es la creación de una herramienta que permitirá escanear el tráfico que pasa por una red.

Otro objetivo, es conocer los principales protocolos de red, analizar los paquetes de red, para controlar las comunicaciones, comprobar errores, fallos de seguridad, etc.

Se pretende obtener una aplicación con una interfaz amigable con los siguientes requisitos funcionales:

- Visualizar todos los paquetes de red en la ventana principal de la aplicación.
- Filtrar los paquetes por una dirección IP, Puerto o protocolo.
- Guardar en un fichero todos los paquetes capturados, para poder analízalos posteriormente.
- Mostrar datos estadísticos mediante una gráfica circular que permita visualizar el total de paquetes según el protocolo.

1.3 Enfoque y método seguido

El método seguido para realizar la aplicación desarrollada en este trabajo es el modelo espiral.

El modelo de ciclo de vida en espiral, consiste en realizar diversas iteraciones, pasando por cada una de sus fases consecutivamente. A diferencia de otros tipos de métodos, que no tienen vuelta atrás, en el modelo en espiral se pueden hacer las iteraciones que se consideren necesarias y estas son sus fases principales:

1. Objetivo y Determinación de requerimientos. En este paso se definen los objetivos específicos para posteriormente identifica las limitaciones del proceso y del sistema de software.

2. Análisis y evaluación de riesgos. En este paso se efectúa un análisis para los posibles riesgos identificados del proyecto, se definen los pasos a seguir para reducir los riesgos en los diferentes casos de uso.

3. Desarrollo y Pruebas. En este paso, se produce el desarrollo del sistema de software. Se verifica y se valida el código desarrollado de las diferentes funciones del mismo.

4. Planificación. El siguiente ciclo se planifica al final de cada etapa. Si se producen errores, se buscan soluciones. Una vez desarrollada una de la funciones, se continua con la siguiente función repitiendo todo el ciclo.

1.4 Planificación del Trabajo

En este apartado vamos a realizar una planificación de la realización del proyecto. Teniendo en cuenta los objetivos de la realización del trabajo lo dividiremos en las siguientes tareas, especificando la duración de las mismas:

Tarea 1:

Duración: 2 semanas (28 de febrero al 14 de marzo).

Descripción: Búsqueda de información relacionada con el TFG a realizar:

1. Información sobre los protocolos más importantes, enrutamiento, interpretación de encabezados de paquetes, reenvió de puertos, etc.
2. Información y estudio de las diferentes herramientas de monitoreo de red existentes.

Objetivos:

- Profundizar los conocimientos sobre la monitorización de la red.
- Tener una visión general sobre las herramientas de monitoreo más utilizadas actualmente.

Metas:

- Aportar la documentación sobre la información obtenida.
- Estudio de las principales herramientas de monitoreo.

Tarea 2:

Duración: 3 semanas (15 de marzo al 5 de abril).

Descripción: Análisis y diseño de la herramienta a diseñar.

Objetivos:

- Establecer las características que tendrá la herramienta.
- Tener una visión clara sobre las funcionalidades de la herramienta a desarrollar.
- Generar la documentación de análisis.
- Elección de entorno de desarrollo.
- Entrega PEC1

Metas:

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

Tarea 3:

Duración: 11 semanas (16 de marzo al 1 de junio).

Descripción: Implementación de la herramienta y pruebas unitarias.

Objetivos:

- Generar las pruebas unitarias adecuadas para garantizar el correcto funcionamiento.
- Implementar la herramienta en base al diseño obtenido.
- Realización de las pruebas.

Metas:

- Programación de la aplicación.
- Documentación técnica de la herramienta desarrollada, así como los test unitarios realizados.
- Entrega PEC1 (14 Abril)
- Entrega PEC2 (22 Mayo)

Tarea 4:

Duración: 2 semanas (2 de junio al 16 de junio).

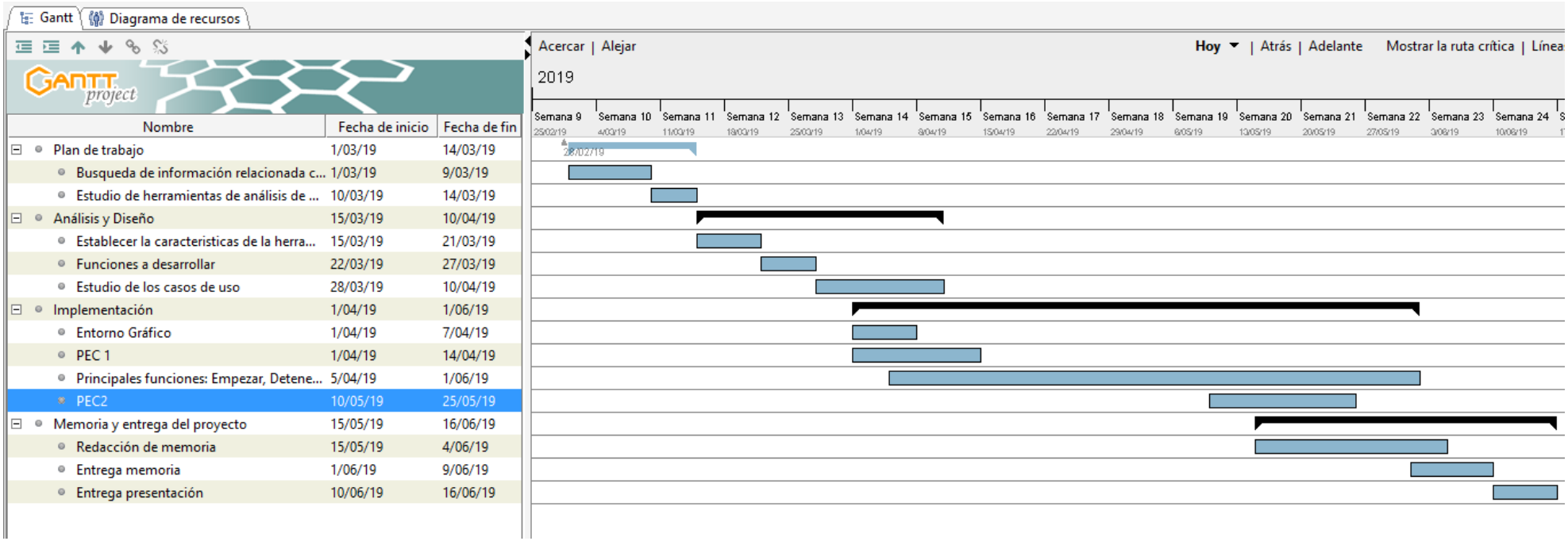
Descripción: Finalización de la memoria y preparación de la presentación de la herramienta desarrollada.

Seguimiento planificación

Se realizó una primera planificación al inicio del trabajo, la cual ha sufrido algunos cambios, debido al tiempo que se le ha dedicado en la implementación de la herramienta. A continuación señalamos algunos cambios:

Desde un principio se dedicó prácticamente todo el tiempo, a la implementación de la aplicación, dedicando menos tiempo a la elaboración de la memoria. Esto conllevo que en la primera entrega, la documentación de análisis, desarrollo e implementación fuera escasa.

En la segunda entrega se presentó una dificultad añadida, al no documentar toda la información a medida que se estaba programando la aplicación, se dedicó demasiado tiempo a la elaboración de la memoria.



Nombre	Fecha de inicio	Fecha de fin
Plan de trabajo	1/03/19	14/03/19
Busqueda de información relacionada c...	1/03/19	9/03/19
Estudio de herramientas de análisis de ...	10/03/19	14/03/19
Análisis y Diseño	15/03/19	10/04/19
Establecer la caracteristicas de la herra...	15/03/19	21/03/19
Funciones a desarrollar	22/03/19	27/03/19
Estudio de los casos de uso	28/03/19	10/04/19
Implementación	1/04/19	1/06/19
Entorno Gráfico	1/04/19	7/04/19
PEC 1	1/04/19	14/04/19
Principales funciones: Empezar, Detene...	5/04/19	1/06/19
PEC2	10/05/19	25/05/19
Memoria y entrega del proyecto	15/05/19	16/06/19
Redacción de memoria	15/05/19	4/06/19
Entrega memoria	1/06/19	9/06/19
Entrega presentación	10/06/19	16/06/19

1.5 Breve sumario de productos obtenidos

A la finalización del trabajo se habrán creado los siguientes entregables:

1. **Memoria**, en formato PDF, documento que detalla toda la información relacionada con el proyecto.
2. **Instalador de la aplicación**.
3. **Código fuente** comentado de la herramienta propuesta.
4. **Video demostrativo del funcionamiento de la herramienta,** explicando los detalles más importantes.

1.6 Breve descripción de los otros capítulos de la memoria

En el segundo capítulo, se hace una investigación preliminar de la información requerida para poder realizar la herramienta. Se hace un estudio del modelo OSI y TCP/IP y de los principales protocolos de red. También se realiza un análisis de dos herramientas actuales de análisis de protocolos, como son, Wireshark y Microsoft Message Analyzer.

En el tercer capítulo, se toman todas las decisiones relacionadas con el diseño conceptual, la definición de los casos de usos, diagrama de clases y la definición de la arquitectura. El resultado de este capítulo será utilizado para realizar la implementación de la herramienta.

En el cuarto capítulo, se realiza la implementación de la herramienta, explicando las principales funciones utilizadas, el desarrollo utilizado, los problemas encontrados y las soluciones utilizadas.

En el quinto capítulo, se explica la funcionalidad de la aplicación a través de un manual de usuario con capturas de pantalla de la aplicación desarrollada.

En el sexto capítulo, se muestra un caso práctico del uso de la aplicación en un equipo del Hospital General de Valencia.

Por último, en el séptimo capítulo, se explica las conclusiones del trabajo, así como, un análisis crítico sobre el logro de los objetivos, del seguimiento de la planificación y las posibles líneas de trabajo futuro.

2. Investigación preliminar

2.1 Modelo OSI

El organismo ISO definió en 1984 un modelo de referencia, llamado modelo de interconexión de sistemas abiertos (OSI), destinado a estandarizar los intercambios de información entre dos máquinas. Se definió lo que debía ser una comunicación de red completa, donde equipos de diferentes fabricantes pudieran comunicarse. Cabe recordar, que en esa época, existían redes propietarias (IBM, DEC, etc.), las cuales usaban protocolos diferentes.

Es una arquitectura basada en capas, se divide en siete capas jerárquicas: Física, Conexión de datos, Red, Transporte, sesión, Presentación y Aplicación, como podemos observar en la Figura 1.

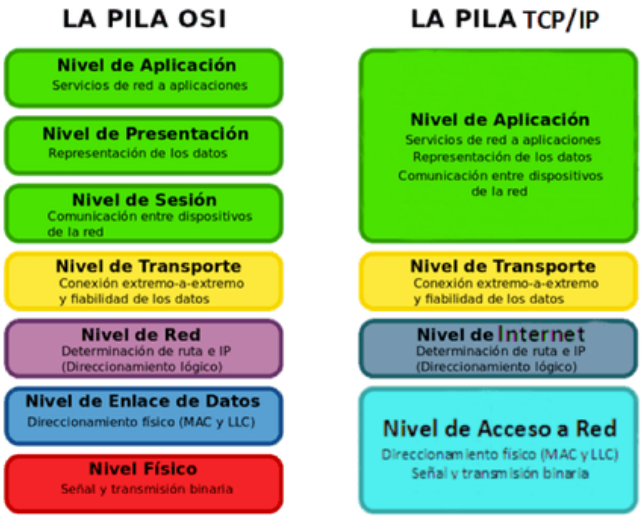


Ilustración 1. Modelo OSI vs TCP/IP

Figura 1. Modelo OSI vs TCP/IP. Fuente: http://domoticaparatodos.com/2016/11/capas-en-los-protocolos/

La comunicación entre ordenadores es bidireccional, el conjunto de capas de red describen caminos tanto para construir y deconstruir paquetes. Cada capa, y cada protocolo específico, debe ser capaz de realizar su función en ambas direcciones.

Cada capa dispone de funcionalidades que le son propias y presta servicio a las capas inmediatamente adyacentes. Aunque el modelo OSI se utiliza muy poco, sirve de referencia para definir el nivel de funcionamiento de un componente de red. Actualmente, el modelo TCP/IP se utiliza de forma más generalizada.

2.2 Modelo TCP/IP

La red ARPANET, nombre de la organización militar *Advanced Research Projects Agency* (ARPA), nació en 1969. Fue creada por el *Department of Defense* (DoD) de Estados Unidos, para conectar diferentes sitios informatizados. ARPA investigó en nuevos protocolos que permitieran conectar redes, fruto de estas investigaciones, nació TCP/IP. En 1976, ARPANET migra a TCP/IP.

En la actualidad, TCP/IP es utilizado por cualquier dispositivo que quiera conectarse a Internet.

La familia TCP/IP, define un modelo en cuatro capas de red: Interfaz de red, Internet, Transporte y Aplicación, como podemos observar en la figura 1.

Podemos dividirla en dos grandes bloques, un primer bloque que contiene las capas de host, son la capa de aplicación y de transporte, estas capas solo se usan en el host origen y destino. Y un segundo bloque, que contiene capas de red, son la capa de Internet y la de acceso a la red, estas capas son las que permiten el tráfico de red.

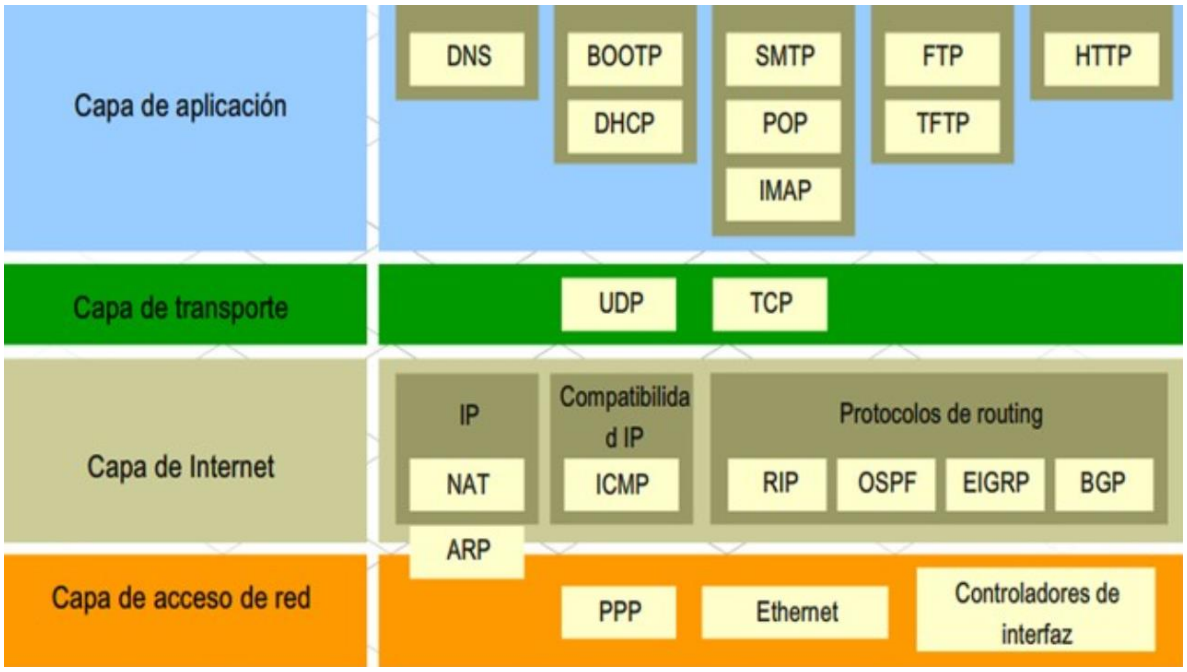


Ilustración 2. Modelo TCP/IP

En primer lugar, vamos a ver la capa de acceso a la red, esta capa equivale a las capas física y conexión de datos del modelo OSI. Ofrece la capacidad de acceder a cualquier red física, es decir, brinda los recursos que se deben implementar para transmitir datos a través de la red local. Por tanto, la capa de acceso a la red contiene especificaciones relacionadas con la transmisión de datos por una red física (red local) Ethernet, en anillo, FDDI, etc.

En este nivel se definen direcciones físicas o direcciones MAC de los dispositivos. Las tramas emplearán estas direcciones para especificar origen y destino de los datos que transportan. Las direcciones MAC están compuestas por 6 bytes (48 bits) donde los tres primeros bytes identifican al fabricante de la tarjeta de red. Los otros tres bytes identifican a la tarjeta.

En segundo lugar, encontramos la capa de Internet, esta capa equivale a la capa de red del modelo OSI. Permite la comunicación de paquetes a través de la red. En esta capa encontramos protocolos, como **IP** (Internet Protocol) existen dos versiones, IPV4 e IPV6, **ARP** (Address Resolution Protocol) y **ICMP** (Internet Control Protocol).

En tercer lugar, encontramos la capa de transporte, esta capa equivale a la capa con el mismo nombre del modelo OSI. Nos permite la transmisión de datos de forma óptima y libre de errores entre dos aplicaciones. En esta capa podemos encontrar el protocolo **TCP** y el **UDP**.

Por último, vamos a ver la capa de aplicación, esta capa equivale a las capas sesión, presentación y aplicación del modelo OSI. Nos permite dar servicios específicos a los usuarios, como correo electrónico, mensajería, etc. En esta capa aparecen diferentes protocolos:

- **FTP** (File Transfer Protocol) para transferencia de archivos.
- **DNS** (Domain Name System).
- **DHCP** (Dynamic Host Configuration Protocol).
- **HTTP** (HyperText Transfer Protocol) para acceso a páginas web.
- **HTTPS** (Hypertext Transfer Protocol Secure)
- **POP3** (Post Office Protocol) para recuperación de correo electrónico.
- **SMTP** (Simple Mail Transport Protocol) para envío de correo electrónico.
- **IMAP** (Internet Message Access Protocol)

2.3 Protocolos

A continuación, vamos a ver que son los protocolos, los principales protocolos que existen y que vamos a utilizar en nuestra aplicación.

Los protocolos son el conjunto de reglas usadas por los dispositivos para poder comunicarse entre sí. Lo protocolos pueden ser implementados por hardware, por software, o por una combinación de ambos.

En primer lugar, veremos los protocolos de la capa de internet, **IP** (Internet Protocol), **ARP** (Address Resolution Protocol), **ICMP** (Internet Control Message Protocol) e **IGMP** (Internet Group Management Protocol).

El protocolo de la capa de internet, **IP** (Internet Protocol), es el encargado del transporte de paquetes desde el origen hasta el destino en una comunicación.

Sus principales características son:

- No orientado a conexión, no existe una previa conexión entre el origen y el destino. Cuando un equipo quiere enviar un paquete a otro equipo, envía directamente este paquete a través de la red.

- No fiable, este protocolo no tiene ningún mecanismo para asegurar que el paquete enviado ha llegado a su destino correctamente.

- Basado en datagrama, un datagrama IP está formado por una cabecera y los propios datos que se quieren enviar. Esta cabecera contiene información del propio protocolo, tamaño, tipo de servicio, etc.

- Existen dos versiones del protocolo IP, el IPv4 y el IPv6. El IPv4 está formado por 4 bytes (32bits) y el IPv6 por 16 bytes (128bits)

El protocolo de resolución de direcciones, **ARP** (Address Resolution Protocol), determina la dirección MAC (dirección física) de un nodo a partir de su dirección IP (dirección lógica), en IPv4. El protocolo RARP (Reverse ARP), hace la resolución inversa.

El protocolo de mensajes de control de Internet, **ICMP** (Internet Control Message Protocol), detecta y registra las condiciones de error de la red. Trabaja en paralelo con el protocolo IP, ya que IP no proporciona mecanismos de control. Por lo que se hizo necesario implementar ICMP para el control de fallos y problemas.

Los principales mensajes que se pueden enviar son:

- Confirmación de host (Echo Request / Echo Response), estos dos mensajes permiten probar si un nodo se puede comunicar con otro.
- Destino inalcanzable (Destination Unreachable), avisa que no se ha podido enrutar el paquete hacia su destino.
- Tiempo excedido (Time Exceeded), Este mensaje indica que se ha sobrepasado el tiempo de espera para el destinatario.
- Redireccionamiento de ruta (Redirect), indica al emisor que existe un mejor camino hacia el destino.
- Disminución de velocidad en origen (Source Quench), permite implementar un mecanismo básico de control de flujo.

El protocolo de red **IGMP** se utiliza para intercambiar información acerca del estado de pertenencia entre enrutadores IP que admiten la multidifusión y miembros de grupos de multidifusión. Los hosts miembros individuales informan acerca de la pertenencia de hosts al grupo de multidifusión y los enrutadores de multidifusión sondean periódicamente el estado de la pertenencia.

En segundo lugar, veremos los protocolos de la capa de transporte, **TCP** y **UDP**. Estos protocolos son utilizados para transmitir datos a través de redes basadas en IP.

UDP (User Datagram Protocol) es un protocolo no orientado a conexión documentado en el RFC 768 de la Internet Engineering Task Force (IETF). Cuando el emisor envía datos al receptor, el flujo es unidireccional, el receptor no envía confirmación al emisor, ni el emisor conecta previamente con el receptor. Es más rápido pero menos fiable que el protocolo TCP. No asegura la conexión, el proceso de comprobación, lo salta a la capa superior, la capa de aplicación.

Se utiliza principalmente para protocolos donde existe un gran intercambio de paquetes de conexión/desconexión, como DHCP, BOOTP, DNS. También es utilizado para transmitir audio y video en tiempo real.

TCP (Transmission Control Protocol) es un protocolo orientado a conexión, luego es el responsable de que se produzca correctamente la transmisión.

Sus principales características son que permite poner los datagramas en orden de nuevo cuando proceden del protocolo IP, permite el monitoreo del flujo de datos y de esta forma evitar la saturación de la red, admite que se formen los datos en segmentos de longitud variada para entregarlos al protocolo IP, permite comenzar y finalizar la comunicación y posibilita la realización de la tarea de multiplexar/demultiplexar, transmitir datos desde diversas aplicaciones en la misma línea.

A partir del uso del protocolo TCP, las aplicaciones pueden comunicarse de forma segura independientemente de las capas inferiores, gracias al sistema de acuse de recibo de dicho protocolo.

Por último, veremos algunos protocolos de la capa de aplicación:

FTP, protocolo de red utilizado para la transferencia de archivos de un sistema a otro conectados a una red TCP, basado en la arquitectura cliente-servidor y de una forma fiable y eficiente.

DNS, base de datos distribuida y jerárquica, que almacena la información necesaria para los nombres de dominio. Sus usos principales son la asignación de nombres de dominio a direcciones IP y la localización de los servidores de correo electrónico correspondientes para cada dominio.

DHCP, protocolo cliente/servidor que permite a los clientes de una red TCP/IP obtener información para su configuración. Los mensajes DHCP utilizan el puerto 67 (UDP) como puerto del servidor y el puerto 68 (UDP) como puerto del cliente.

HTTP, protocolo de comunicación que permite la transferencia de información en Internet. Los datos que se solicitan a través de HTTP viajan desde el servidor al cliente en texto plano, sin cifrar ni asegurar. Utiliza el puerto 80 para las conexiones.

HTTPS, protocolo de transferencia de hipertexto seguro. HTTPS utiliza TLS, (Transport Layer Security), protocolo que nos brinda mayor seguridad y mayor privacidad en nuestras conexiones. Utiliza el puerto 443 para las conexiones.

POP3, protocolo utilizado en Internet para la recepción de mensajes de correo almacenados en un servidor. POP3 administra la autenticación utilizando el nombre de usuario y la contraseña. Sin embargo, esto no es seguro, ya que las contraseñas, al igual que los correos electrónicos, circulan por la red como texto sin codificar.

SMTP, protocolo de red utilizado para el intercambio de mensajes de correo electrónico entre computadoras u otros dispositivos. Se asocia con otros protocolos, como el POP o IMAP, otorgando a SMTP la tarea específica de enviar correo, y el de recibirlos a los otros protocolos, POP O IMAP.

IMAP, protocolo que permite el acceso a mensajes de correo electrónico que se encuentran almacenados en un servidor desde cualquier ordenador que tenga conexión a Internet. IMAP es un protocolo de acceso al correo electrónico, pero no es un protocolo de envío de email.

2.4 Encapsulamiento de paquetes

El paquete es la unidad de información básica que se transfiere a través de una red. El paquete básico se compone de un encabezado con las direcciones de los sistemas de envío y recepción, y un cuerpo, o carga útil, con los datos que se van a transferir. Cuando el paquete se transfiere a través de la pila de protocolo TCP/IP, los protocolos de cada capa agregan o eliminan campos del encabezado básico. Cuando un protocolo del sistema de envío agrega datos al encabezado del paquete, el proceso se denomina encapsulado de datos.

En la siguiente imagen podemos ver, el ciclo de vida de un paquete básico empieza cuando se envía un paquete o mensaje, y finaliza cuando la aplicación adecuada del sistema receptor recibe el paquete.

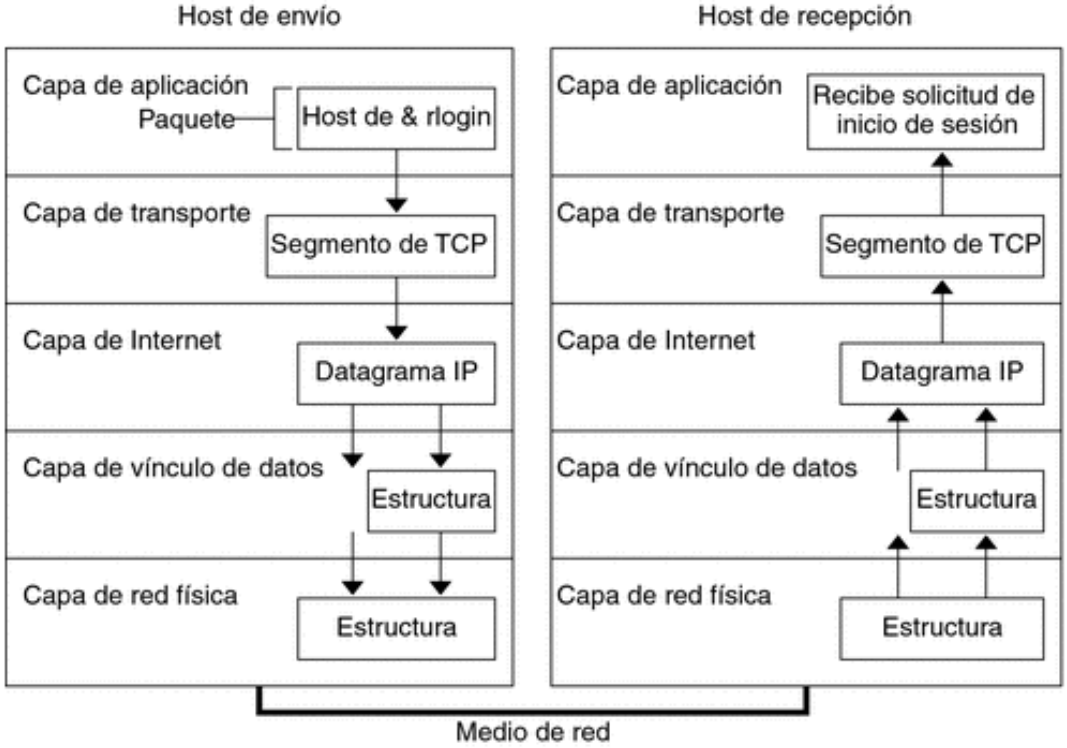


Ilustración 3. Transferencia de un paquete a través de la pila TCP/IP

2.5 Analizadores de paquetes en el mercado

Un sniffer, es un analizador de paquetes, un programa de captura de paquetes, usado para controlar y analizar el tráfico de red. El sniffer captura cada paquete, lo codifica y a continuación muestra su información para ser analizada. Es utilizado para generar información de errores, monitorización, detectar intentos de intrusión en la red, fallos de seguridad, etc.

Se utilizan para descubrir problemas en la red, encontrar cuellos de botella en algún lugar de la red, detección de intrusos, analizar la información que se trasmite por la red, convertir este tráfico en un formato inteligible, etc.

Los principales analizadores de red y sniffers que he analizado son:

2.5.1 Wireshark

Es el analizador de protocolos de red más importante y más utilizado del mundo. Su principal objetivo es el análisis de tráfico, pero además es una excelente aplicación didáctica para el estudio de las comunicaciones y la resolución de problemas de red.

Dispone de una interfaz sencilla e intuitiva que permite desglosar por capas cada uno de los paquetes capturados.

Cuando iniciamos el programa, para comenzar a capturar paquetes, hacemos doble clic en la interfaz que deseemos. En mi equipo, seleccione, la conexión de área local 2.

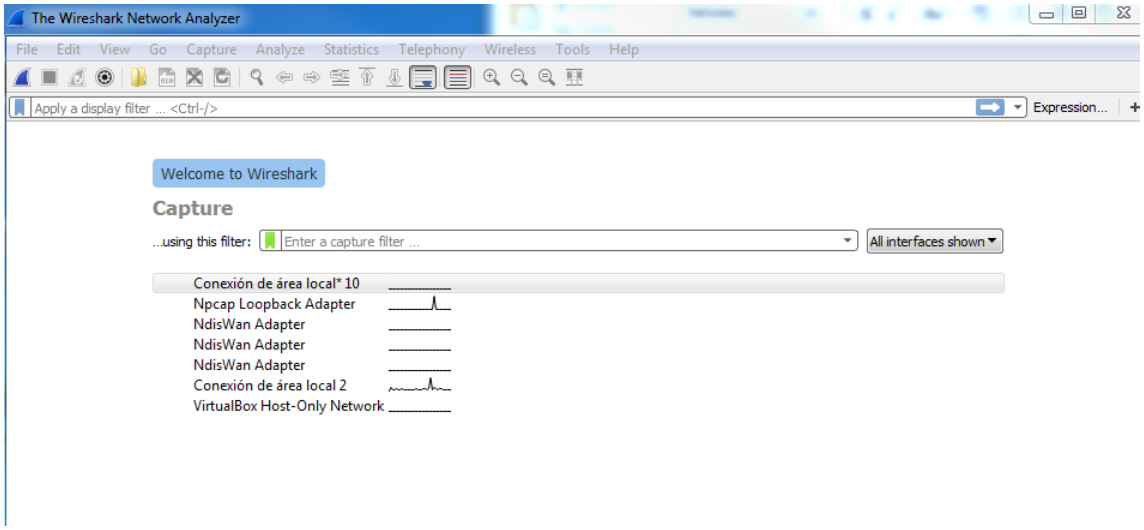


Ilustración 4. Captura 1 - Wireshark

Para comprobar que está funcionando correctamente, hago pings continuos desde mi máquina (172.17.183.210), a un servidor del hospital (10.192.152.55). Como podemos observar en la siguiente captura, vemos diferentes paquetes de ICMP, y en el campo “Info”, los diferentes pings realizados.

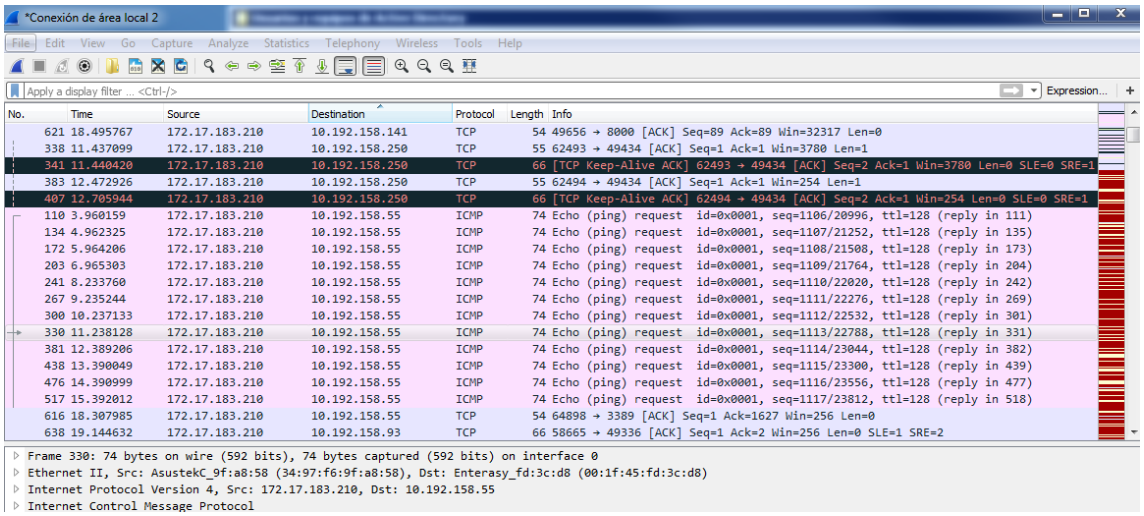


Ilustración 5. Captura 2 - Wireshark

Una vez capturado los paquetes podemos realizar diferentes filtros para buscar protocolos específicos, o podemos seguir una secuencia TCP con la opción Follow\ TCP Stream. En la siguiente captura, he realizado botón derecho sobre un paquete TCP para poder ver la secuencia completa entre mi equipo y el servidor.

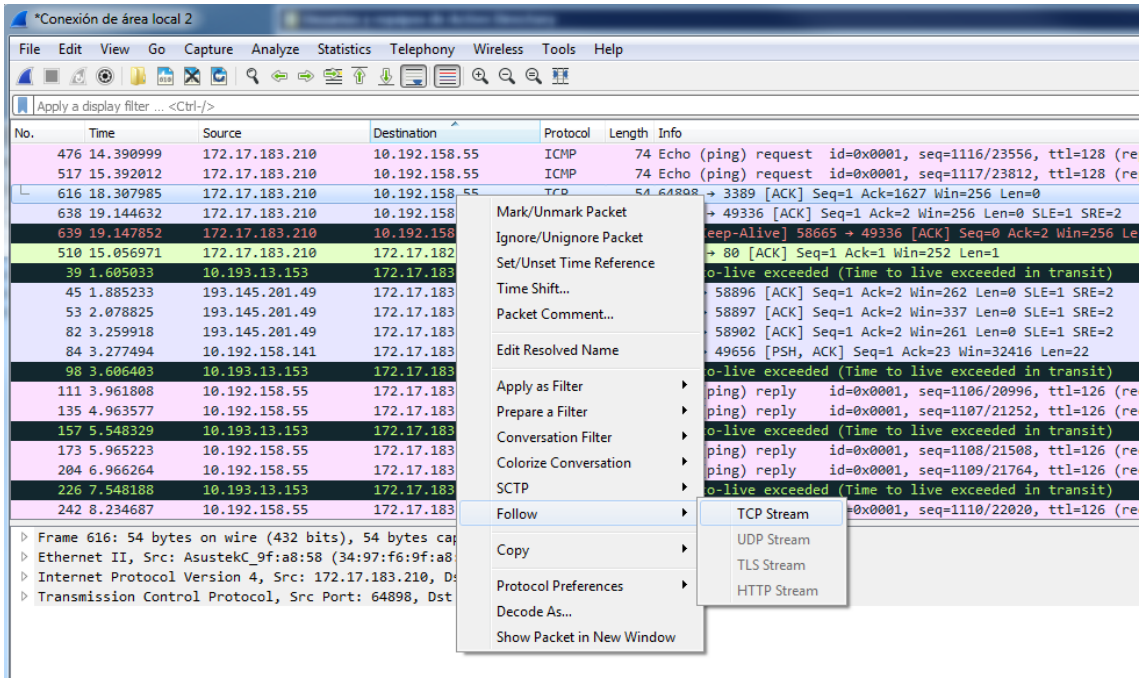


Ilustración 6. Captura 3 - Wireshark

En la siguiente captura, se observa la conversación completa entre mi equipo y el servidor. Se utiliza el protocolo criptográfico TLSv1, que proporciona comunicaciones seguras en una red.

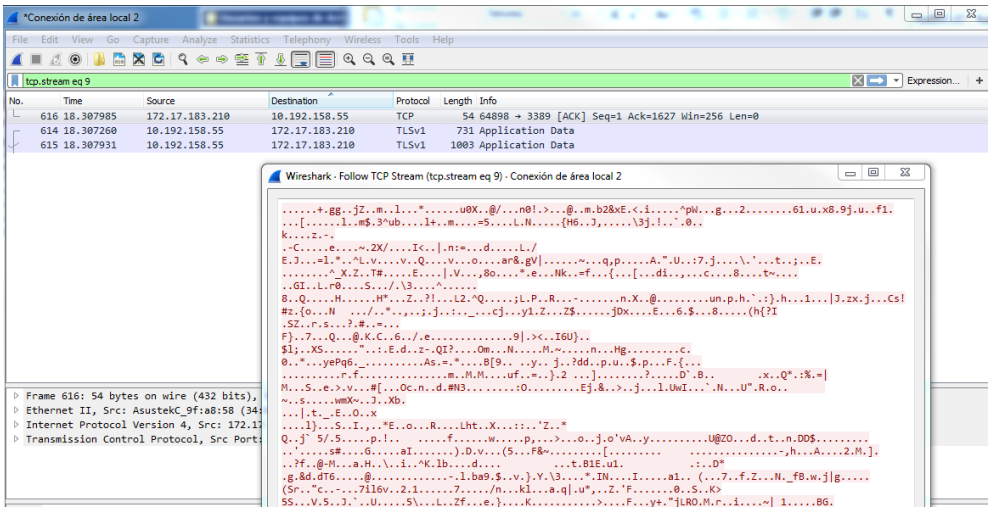


Ilustración 7. Captura 4 - Wireshark

A nivel estadístico, la aplicación Wireshark proporciona un rango amplio de estadísticas de red, podemos acceder a ellas desde el menú Statistics, en la siguiente captura se muestra las estadísticas de información general de los paquetes capturados según los protocolos. La opción es Protocol Hierarchy.

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	22125	100.0	11473164	293 k	0	0	0
Ethernet	100.0	22125	2.7	309750	7928	0	0	0
Logical-Link Control	0.7	161	0.1	16971	434	0	0	0
Spanning Tree Protocol	0.7	155	0.1	15810	404	155	15810	404
Cisco Discovery Protocol	0.0	6	0.0	648	16	6	648	16
Link Layer Discovery Protocol	0.0	6	0.0	870	22	6	870	22
InterSwitch Message Protocol	0.0	5	0.0	735	18	5	620	15
Internet Protocol Version 6	1.8	394	0.1	15760	403	0	0	0
User Datagram Protocol	1.1	254	0.0	2032	52	0	0	0
Multicast Domain Name System	0.0	3	0.0	120	3	3	120	3
Link-local Multicast Name Resolution	0.2	38	0.0	976	24	38	976	24
DHCPv6	0.9	204	0.2	19423	497	204	19423	497
Data	0.0	9	0.1	6507	166	9	6507	166
Internet Control Message Protocol v6	0.6	140	0.0	4288	109	140	4288	109
Internet Protocol Version 4	79.3	17539	3.1	352364	9019	0	0	0
User Datagram Protocol	28.1	6216	0.4	49728	1272	0	0	0
Simple Service Discovery Protocol	3.9	856	1.5	169107	4328	856	169107	4328
Simple Network Management Protocol	0.2	40	0.1	6390	163	40	6390	163
Service Location Protocol	0.0	4	0.0	176	4	4	176	4
NetBIOS Name Service	4.4	965	0.5	53794	1376	965	53794	1376
NetBIOS Datagram Service	0.4	81	0.1	16624	425	0	0	0
SMB (Server Message Block Protocol)	0.4	81	0.1	9310	238	0	0	0
SMB MailSlot Protocol	0.4	81	0.0	2025	51	0	0	0
Microsoft Windows Browser Protocol	0.4	81	0.0	2344	59	81	2344	59
Multicast Domain Name System	0.1	23	0.0	920	23	23	920	23
Link-local Multicast Name Resolution	1.2	273	0.1	7214	184	273	7214	184
Dynamic Host Configuration Protocol	0.4	99	0.3	31202	798	99	31202	798
Dropbox LAN sync Discovery Protocol	0.1	22	0.0	3190	81	22	3190	81
Domain Name System	0.1	18	0.0	1792	45	18	1792	45
Data	17.3	3819	45.6	5227518	133 k	3819	5227518	133 k
Connectionless Lightweight Directory Access Protocol	0.1	16	0.0	2506	64	16	2506	64
Transmission Control Protocol	49.2	10883	43.3	4962265	127 k	6248	2630619	67 k
Transport Layer Security	16.9	3733	41.5	4764133	121 k	3356	3504868	89 k
Tabular Data Stream	0.0	6	0.0	1080	27	6	1080	27
Post Office Protocol	0.4	98	0.1	12884	329	92	9486	242
Internet Message Format	0.0	6	0.0	3398	86	6	3398	86
NetBIOS Session Service	0.4	98	0.2	24861	636	6	178	4
SMB2 (Server Message Block Protocol version 2)	0.2	54	0.1	9640	246	54	9644	246
SMB (Server Message Block Protocol)	0.2	38	0.1	14675	375	31	5367	137
SMB Pipe Protocol	0.0	7	0.0	100	2	1	13	0
Microsoft Windows Lanman Remote API Protocol	0.0	6	0.1	10620	271	6	8914	228
Malformed Packet	0.0	3	0.0	0	0	2	0	0
Lightweight Directory Access Protocol	0.2	34	0.1	11630	297	18	8312	212
Internet Message Access Protocol	0.5	116	0.0	4022	102	116	4022	102
Hypertext Transfer Protocol	18.4	4063	42.0	4822456	123 k	344	49485	1266
Portable Network Graphics	0.0	3	0.0	460	11	3	2499	63
Line-based text data	0.0	7	0.5	56931	1457	7	17701	453
Compuserve GIF	0.0	1	0.0	1251	32	1	1633	41
Distributed Computing Environment / Remote Procedure Call (DCE/RPC)	0.2	52	0.2	20895	534	26	17227	440
DRSUAPI	0.1	12	0.0	1392	35	12	1392	35
DCOM OXID Resolver	0.0	10	0.0	60	1	10	60	1
DCE/RPC Endpoint Mapper	0.0	4	0.0	584	14	4	584	14
Data	2.4	534	0.2	27820	712	534	27820	712

Ilustración 8. Captura 5 - Wireshark

2.5.2 Microsoft Message Analyzer

Herramienta para capturar, mostrar y analizar el tráfico de mensajes de protocolo y otros mensajes del sistema. Message Analyzer también le permite importar, agregar y analizar datos de los archivos de registro y rastreo.

Message Analyzer le permite mostrar datos de seguimiento, registro y otros mensajes en numerosos formatos de visor de datos, incluida una vista de cuadrícula de árbol predeterminada y otras vistas gráficas seleccionables que emplean cuadrículas, gráficos y componentes de visualizador de línea de tiempo que proporcionan resúmenes de datos de alto nivel y otras estadísticas.

También le permite configurar sus propios visores de datos personalizados. Además, Message Analyzer no solo es una herramienta eficaz para solucionar problemas de red, sino también para probar y verificar implementaciones de protocolos.

Instalo la aplicación en mi equipo del hospital y procedo a analizar los paquetes.

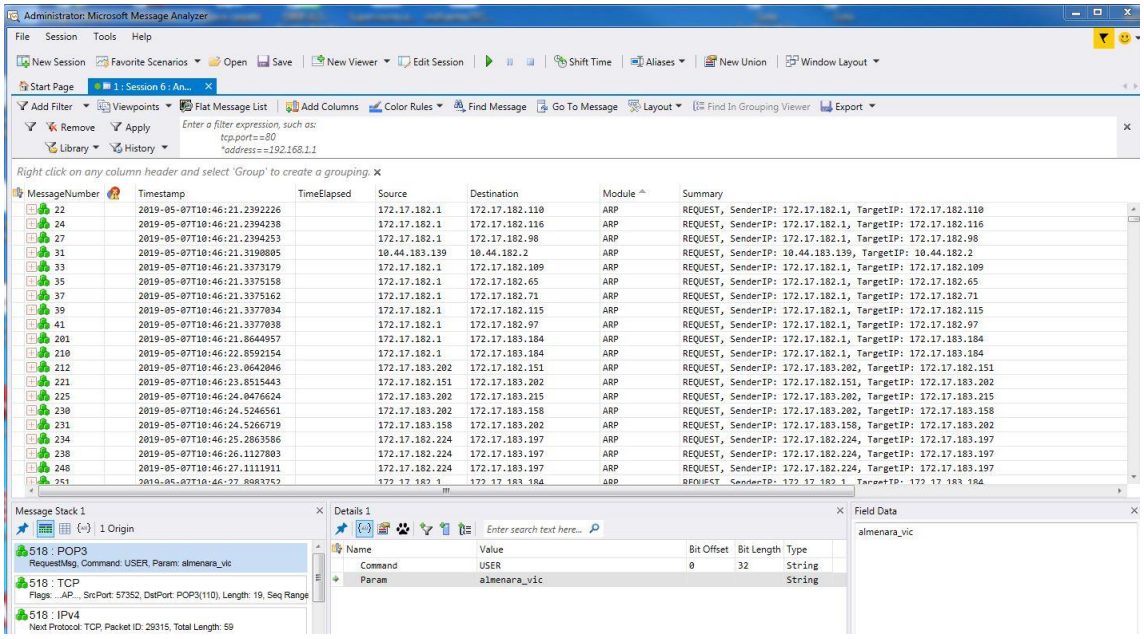


Ilustración 9. Captura 1 - Microsoft Message Analyzer

Revisando los paquetes, compruebo que aparece mi contraseña sin encriptar de mi correo corporativo en un paquete, filtro por el protocolo POP3, para comprobar si sale más información.

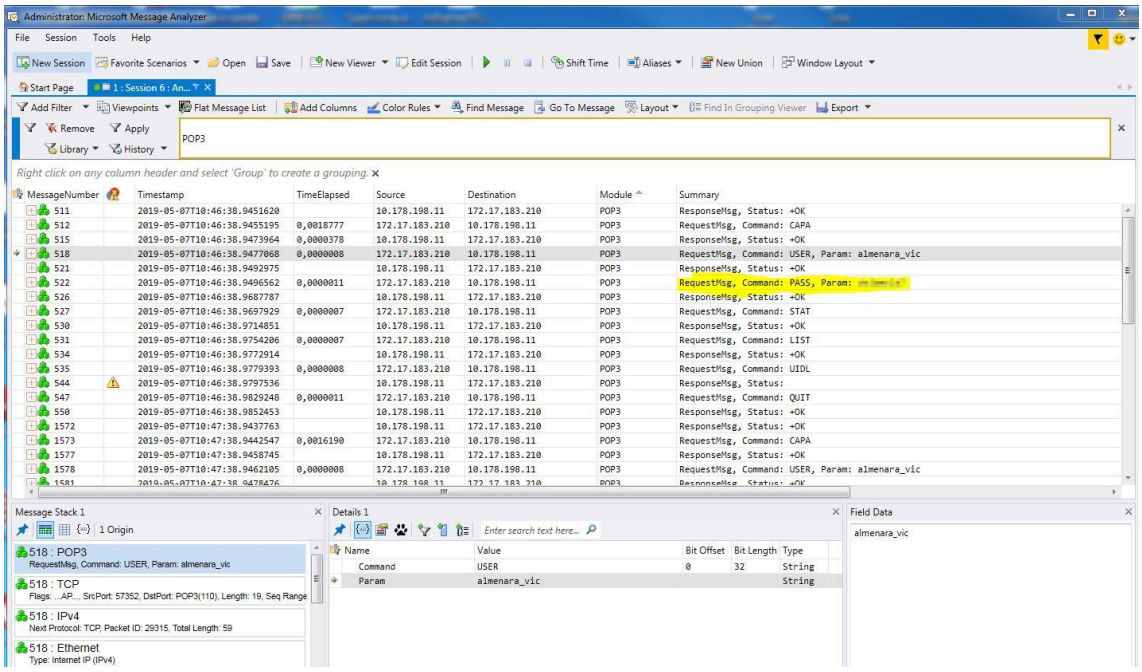


Ilustración 10. Captura 2 - Microsoft Message Analyzer

3. Análisis y Diseño

En este apartado se trata por una parte el análisis, a través de los diferentes requisitos que debe tener la aplicación y por otra parte, se aborda el diseño y la estructura que hemos seguido y utilizado para desarrollar la aplicación de un modo sencillo y modulable. Se especifican las diferentes funciones implementadas, a través de los casos de uso y diagrama de clases.

En primer lugar, procedemos a describir que realizará la aplicación, la herramienta será capaz de capturar todos los paquetes que pasan por la red. Los datos capturados se podrán mostrar en modo hexadecimal y ASCII.

Los requisitos del sistema serán los siguientes:

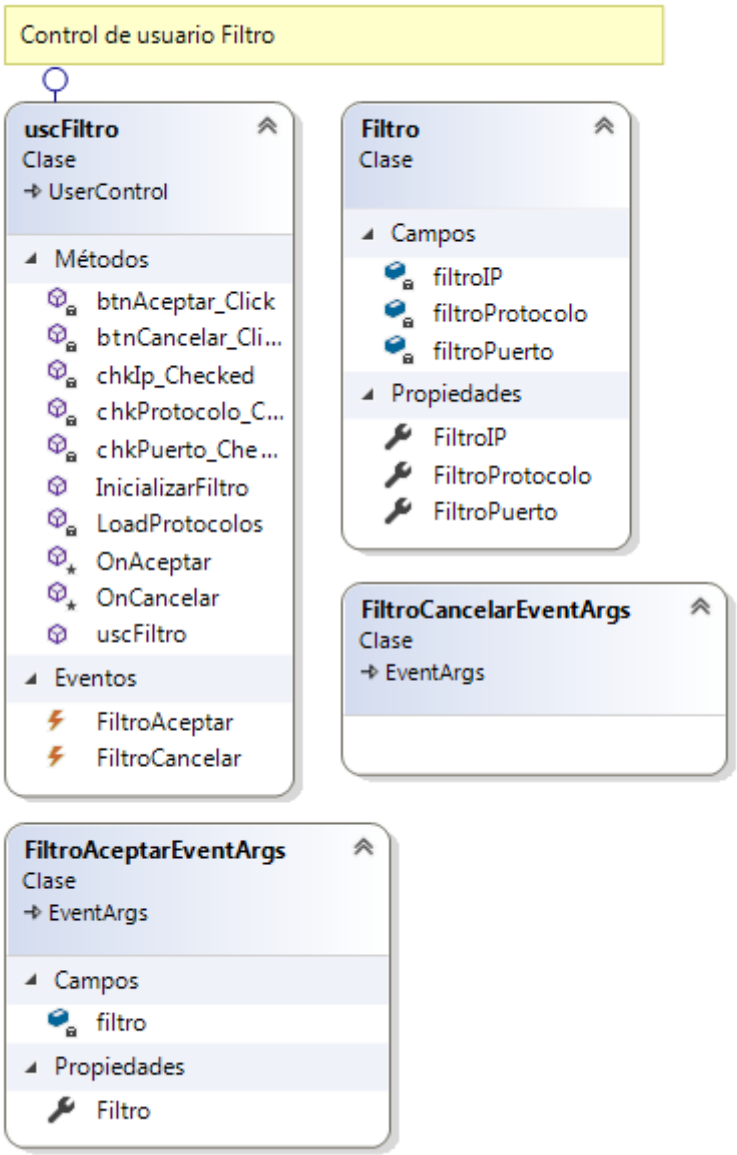
- Para capturar los paquetes que pasa por el dispositivo de red, el sistema deberá permitir poner el dispositivo en modo promiscuo y ejecutarse en modo administrador.
- Se debe poder filtrar por IP, Puerto y por tipo de protocolo.
- Se podrá guardar en un fichero de forma legible todos los paquetes capturados para poder ser analizados posteriormente.
- Para sacar datos estadísticos, se configurará una gráfica circular que permita ver gráficamente los resultados obtenidos.

A continuación, describiremos los diagramas de clases y los casos de uso utilizados para el diseño de la aplicación.

3.1 Diagrama de clases

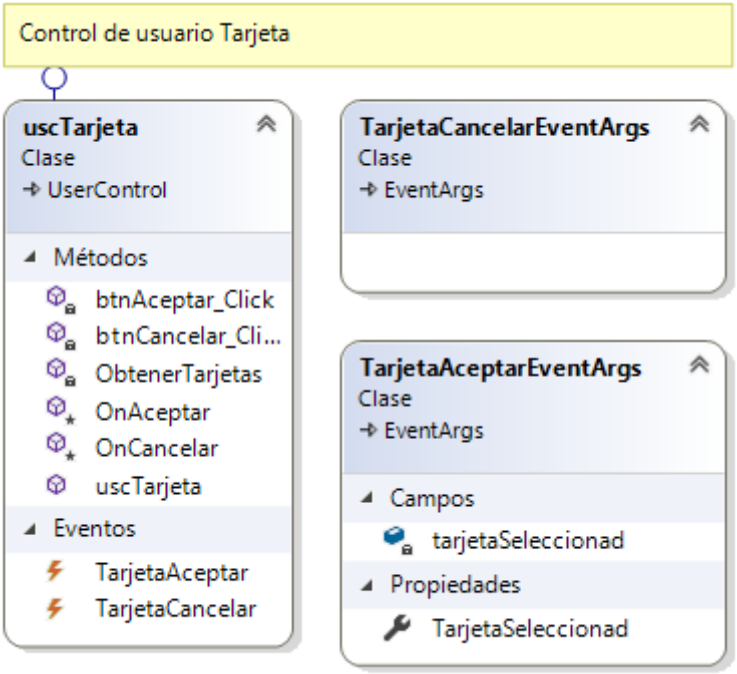
En la siguiente imagen se pueden ver las diferentes clases utilizadas para la implementación del control de filtro.

- Filtro: es la clase que se utiliza para almacenar el filtro. Se aprecian los campos y sus propiedades.
- uscFiltro: es la propia del control de filtro, esta hereda de la clase UserControls del framework. En ella se han creado los métodos referentes a los eventos de los controles de la pantalla, como botones, campos de verificación. Además, de algunos procedimientos que permiten inicializar y cargar desplegables.
- FiltroCanclearEventArgs y FiltroAceptarEventArgs: se han creado en el control dos handlers que permiten capturar los eventos que se disparan para que la pantalla principal los capture y actúe en consecuencia.


Control de usuario Filtro
uscFiltro
Clase
UserControl
Métodos
btnAceptar_Click
btnCancelar_Cli...
chkIp_Checked
chkProtocolo_C...
chkPuerto_Che...
InicializarFiltro
LoadProtocolos
OnAceptar
OnCancelar
uscFiltro
Eventos
FiltroAceptar
FiltroCancelar
Filtro
Clase
Campos
filtroIP
filtroProtocolo
filtroPuerto
Propiedades
FiltroIP
FiltroProtocolo
FiltroPuerto
FiltroCancelarEventArgs
Clase
EventArgs
FiltroAceptarEventArgs
Clase
EventArgs
Campos
filtro
Propiedades
Filtro

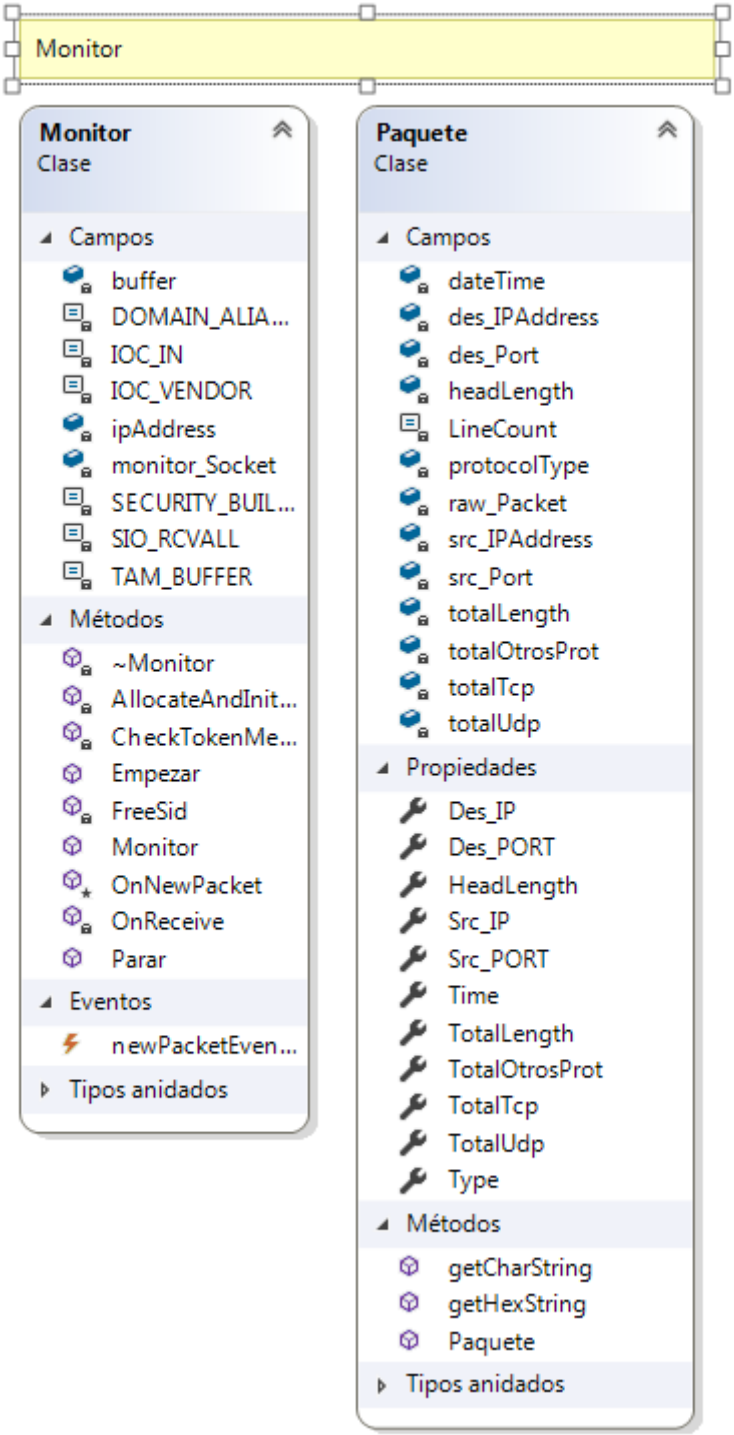
En la siguiente imagen se pueden ver las diferentes clases utilizadas para la implementación del control de selección de tarjeta.

- uscTarjeta: es la propia del control de selección de tarjeta, esta hereda de la clase UserControls del framework. En ella se han creado los métodos referentes a los eventos de los controles de la pantalla, como botones, campos de verificación. Además, de procedimientos que permiten obtener los dispositivos de red.
- TarjetaCanclearEventArgs y TarjetaAceptarEventArgs: se han creado en el control dos handlers que permiten capturar los eventos que se disparan para que la pantalla principal los capture y actúe en consecuencia.



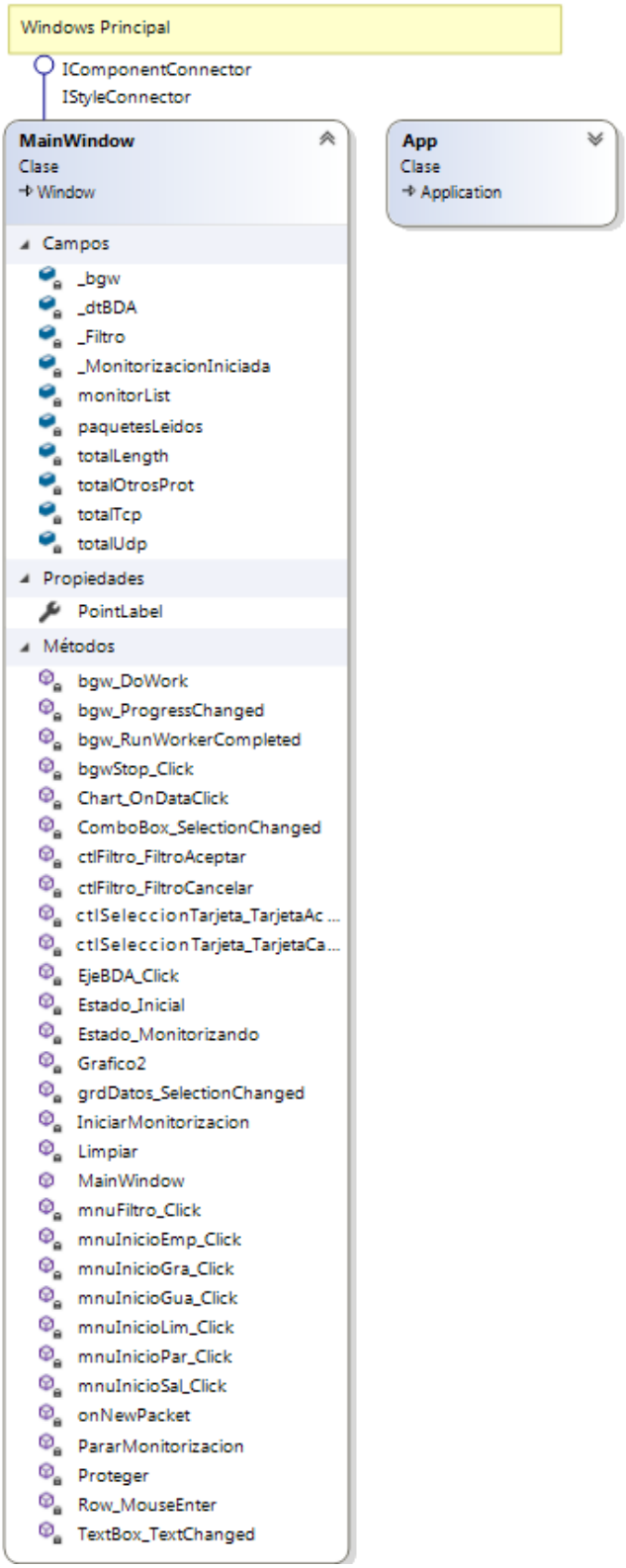
En esta tercera captura de diagrama de clases, encontramos la clase de Monitor, con los diferentes métodos implementados, esta clase permite que iniciar la monitorización de la red por medio del método “Empezar”, este abre un socket en modo promiscuo, esta dispara una serie de eventos como es “OnRecive” que por cada trama crea un paquete.

La clase paquete es una representación de una trama de datos, se han definido las propiedades principales. Cuando se crea el paquete en el constructor se ha desarrollado para transformar los bits de la trama a las propiedades.



En esta última captura de diagrama de clases, encontramos la clase de MainWindow que es la ventana principal de la aplicación, con los diferentes métodos implementados, como son Gráfico2 y los diferentes controles de filtro y tarjeta. Así como todos los eventos que producen los diferentes controles de la ventana.

La clase App se utiliza para aplicaciones desarrolladas en XAML, es el punto de inicio, además de poder definir otras características generales para la aplicación.



3.2 Casos de uso

Los casos de uso formulan las necesidades de los usuarios con el lenguaje de los actores. Los modelos de casos de uso describen los servicios que se esperan del sistema, utilizando para ello la forma de interacciones entre los actores y el sistema.

A continuación, se detallan los casos de uso de las diferentes funciones de la aplicación:

<Identificador>	**CU001**		
Título	**Empezar escaneo de paquetes**		
Descripción	El sistema deberá permitir al usuario iniciar el escaneo de los paquetes que pasan por el dispositivo de red seleccionado, según se describe en el siguiente caso de uso:		
Precondición	Ejecutar la aplicación en modo administrador		
Secuencia Normal	**Paso**	**Acción**	
	1	Pulsar el botón Empezar	
	2	Seleccionar la tarjeta por la que quiere empezar a escanear la red y darle al botón aceptar.	
		2a	Si no se selecciona una tarjeta el sistema deberá mostrar un mensaje indicando que debe seleccionar una tarjeta.
		2b	Si cancela la ventana de seleccionar tarjeta, el sistema deberá cerrar dicha ventana y mostrar la página principal
	3	El programa empieza a capturar los paquetes de red.	
Postcondición	Debe de haber tráfico en la red y el antivirus u otros elementos deben permitir la utilización de la tarjeta en modo promiscuo.		
Excepciones	**Paso**	**Acción**	
	p	En el caso de que el usuario no ejecute la aplicación en modo administrador, el sistema deberá cerrar la aplicación y abrirla en modo administrador.	
	q	En el caso de seleccionar una tarjeta no válida, el sistema deberá indicarlo.	

<Identificador>	**CU002**		
Título	**Filtrado de paquetes**		
Descripción	El sistema deberá permitir al usuario filtrar por IP, puerto o protocolo, según se describe en el siguiente caso de uso:		
Precondición	Debe marcar el Checkbox correspondiente.		
Secuencia Normal	**Paso**	**Acción**	
	1	Pulsar el botón FIltro	
	2	Marcar el checbox que desee el usuario	
	3	Indicar o seleccionar el filtrado deseado	
		2a	Si se pulsa el botón aceptar el sistema deberá filtrar los paquetes capturados
		2b	Si se pulsa el botón cancelar] el sistema deberá cerrar la ventana cancelar.
	4	Se muestran los paquetes filtrados.	
Excepciones	**Paso**	**Acción**	
	p	En el caso de que la dirección IP o puerto no se hayan escrito correctamente, el sistema deberá informar al usuario para su corrección.	
	q	Si no selecciona ningún protocolo, el sistema deberá mostrar un mensaje indicándolo al usuario.	

<Identificador>	**CU003**		
Título	**Guardar paquetes**		
Descripción	El sistema deberá permitir al usuario guardar en un fichero los paquetes capturados, según se describe en el siguiente caso de uso:		
Precondición	Previamente haber realizado un escaneo de la red.		
Secuencia Normal	**Paso**	**Acción**	
	1	Pulsar el botón Guardar	
	2	Revisar si se ha realizado el escaneo previamente.	
		2a	Si, solicitar la ubicación y el nombre del fichero.
		2b	No, mostrar un mensaje al usuario.
	3	Guardar los paquetes en el fichero.	
Excepciones	**Paso**	**Acción**	
	p	En el caso de que no se pueda guardar el fichero, comunicar al usuario el problema.	

<Identificador>	**CU004**		
Título	**Gráfica de paquetes**		
Descripción	El sistema deberá permitir al usuario mostrar una estadística en formato de gráfica circular con la información capturada, según se describe en el siguiente caso de uso:		
Precondición	Previamente haber realizado un escaneo de la red o estar escaneando.		
Secuencia Normal	**Paso**	**Acción**	
	1	Pulsar el botón Gráfica.	
	2	Revisar si se ha realizado el escaneo previamente.	
		2a	Si, Mostrar el gráfico.
		2b	No, mostrar un mensaje al usuario.

4. Implementación

El lenguaje utilizado para la realización de la aplicación es C#, un lenguaje de programación orientado a objetos diseñado por la compañía Microsoft para la plataforma .NET. Presenta los atributos positivos de C++, Java y Visual Basic y los mejora otorgando un lenguaje fuerte y actualizado. Mediante el lenguaje C# podemos desarrollar todo tipo de productos, desde aplicaciones de gestión para escritorio, para la web, programas de terminal, juegos, aplicaciones móviles, etc.

Sus principales características son:

- **Sintaxis sencilla** que facilita al desarrollador la escritura de código.
- **Sistema de tipo unificado**, permitiendo realizar operaciones comunes y que los valores de todos los tipos se puedan almacenar, transportar y utilizar de manera coherente.
- **Bibliotecas**. Todos los compiladores de C# tienen un mínimo de biblioteca de clases disponibles para usar.
- **Integración con otros lenguajes**. Tanto con sus antecesores como con Java y muchos otros lenguajes de programación

El entorno que se utilizará para realizar la herramienta es Microsoft Visual Studio 2017. El entorno de desarrollo integrado de Visual Studio es un panel de inicio creativo que se puede usar para editar, depurar y compilar código y, después, publicar una aplicación. Incluye compiladores, herramientas de finalización de código, diseñadores gráficos y muchas más características para facilitar el proceso de desarrollo de software.

Con este trabajo, se pretende profundizar en el conocimiento de este lenguaje, ya que es uno de los más utilizados en el área de desarrollo de mi actual lugar de trabajo.

A continuación, vamos explicar la implementación de las principales funciones de la aplicación:

4.1 Capturar los paquetes

Para capturar los paquetes he utilizado el espacio de nombres System.Net.Sockets, con clases como socket, IPAddress y sus diferentes enumeraciones. Tras una búsqueda exhaustiva, las principales referencias que se han usado como soporte son:

https://docs.microsoft.com/es-es/dotnet/api/system.net.sockets?view=netframework-4.8

http://es.tldp.org/Tutoriales/PROG-SOCKETS/prog-sockets.html

https://es.wikibooks.org/wiki/Programaci%C3%B3n_en_C/Sockets#socket();

Se ha utilizado diversas clases ya contenidas en el framework 4.8 de Microsoft .Net. Entre ellas:

- IPEndPoint: Permite representar un dirección de red como un recurso.
- Socket: es una clase que encapsula la funcionalidad de conectividad entre dos elementos con diferentes direcciones de red. Encapsula todo el flujo y los datagramas.

En las siguientes capturas podemos ver parte de la implementación:

En primer lugar, se definen el socket, la dirección IP y el array para almacenar los paquetes capturados. A continuación, se implementa “Empezar monitorización”, donde se comprueba si la dirección es IPv4 o IPv6, y se abre el socket por donde va a empezar a capturar en modo promiscuo.

```
private Socket monitor_Socket;
private IPAddress ipAddress;
private byte[] buffer;

1 referencia
public Monitor(IPAddress ip)
{
    this.ipAddress = ip;
    this.buffer = new byte[TAM_BUFFER];
}

0 referencias
~Monitor()...

#region Empezar monitorización

1 referencia
public void Empezar()
{
    if (monitor_Socket == null)
    {
        try
        {

            if (ipAddress.AddressFamily == AddressFamily.InterNetwork)
            { // IPv4
                monitor_Socket = new Socket(AddressFamily.InterNetwork, SocketType.Raw, System.Net.Sockets.ProtocolType.IP);
            }
            else
            { // IPv6
                monitor_Socket = new Socket(AddressFamily.InterNetworkV6, SocketType.Raw, System.Net.Sockets.ProtocolType.IP);
            }
            // abre el socket
            monitor_Socket.Bind(new IPEndPoint(ipAddress, 0));
            // establece los modos operativos a bajo nivel para los sistemas operativos
            monitor_Socket.IOControl(SIO_RCVALL, BitConverter.GetBytes((int)1), null);
            //  se indica donde se va a realizar la recepción
            monitor_Socket.BeginReceive(buffer, 0, buffer.Length, SocketFlags.None, new AsyncCallback(this.OnReceive), null);
        }
        catch (Exception e)
        {
            monitor_Socket.Close();
            monitor_Socket = null;
            Console.WriteLine(e.ToString());
        }
    }
}
#endregion
```

Ilustración 11. Captura 1 - Captura de paquetes

Cuando se crea el paquete se obtienen los bytes de las posiciones concretas de la trama para obtener los flags, ip de origen y destino, puerto origen y destino, etc.

```
public Paquete(byte[] raw)
{
    ///No se ha recibido datos se eleva una excepcion
    if (raw == null)
        throw new ArgumentNullException();

    ///Si la longitud es menor que 20, no es correcto;
    if (raw.Length < 20)
        throw new ArgumentException();
    raw_Packet = raw;
    dateTime = DateTime.Now;///La hora en que se capturó el paquete;

    packetlenght = raw.Length;

    ///Obtener longitud de cabecera en el paquete;
    headLength = (raw[0] & 0x0F) * 4;
    if ((raw[0] & 0x0F) < 5)
        throw new ArgumentException(); /// Cabecera no es correcta porque la longitud es incorrecta;

    ///La longitud actual es la misma que se eindica en el flag de la cabecera?
    if ((raw[2] * 256 + raw[3]) != raw.Length)
        throw new ArgumentException(); // La longitud es incorrecta;

    ///Obtener tipo de paquete;
    if (Enum.IsDefined(typeof(ProtocolType), (int)raw[9]))
        protocolType = (ProtocolType)raw[9];
    else
        protocolType = ProtocolType.OTHERS;

    src_IPAddress = new IPAddress(BitConverter.ToUInt32(raw, 12)); // ip origen
    des_IPAddress = new IPAddress(BitConverter.ToUInt32(raw, 16)); // ip destino
    totalLength = raw[2] * 256 + raw[3];

    #region Método para paquetes TCP o UDP;
    // dependiendo del protocolo rcuenta diferente el tamaño
    if (protocolType == ProtocolType.TCP || protocolType == ProtocolType.UDP)
    {
        src_Port = raw[headLength] * 256 + raw[headLength + 1];
        des_Port = raw[headLength + 2] * 256 + raw[headLength + 3];
        if (protocolType == ProtocolType.TCP)
        {
            totalTcp += 1;
            headLength += 20;
        }
        else if (protocolType == ProtocolType.UDP)
        {
            totalUdp += 1;
            headLength += 8;
        }
    }
    else
    {
        totalOtrosProt += 1;
        src_Port = -1;
        des_Port = -1;
    }
    #endregion
}
```

Ilustración 12. Captura 2 - Captura de paquetes

En la siguiente captura, se puede observar el evento OnReceive que se dispara cada vez que se recibe un paquete por la interfaz de red seleccionada.

1. En primer lugar, se detiene la recepción y se realiza una copia del buffer.
2. Con los datos copiados del buffer se crea una clase paquete.
3. Se dispara el evento OnNewPacket que permite que la pantalla se actualice con el nuevo paquete recibido.
4. Se vuelve a iniciar la captura de datos.

```
private void OnReceive(IAsyncResult ar)
{
    try
    {
        int len = monitor_Socket.EndReceive(ar); // se para la recepción de datos
        if (monitor_Socket != null)
        {
            byte[] receivedBuffer = new byte[len];
            Array.Copy(buffer, 0, receivedBuffer, 0, len); // se realiza una copia del buffer para no perderlos con la siguiente recepción
            try
            {
                Paquete packet = new Paquete(receivedBuffer); // se crea una clase con los datos recibidos en el socket
                OnNewPacket(packet); // refresco de los datos en la pantalla
            }
            catch (ArgumentNullException ane)
            {
                Console.WriteLine(ane.ToString());
            }
            catch (ArgumentException ae)
            {
                Console.WriteLine(ae.ToString());
            }
        }
        monitor_Socket.BeginReceive(buffer, 0, buffer.Length, SocketFlags.None, new AsyncCallback(this.OnReceive), null); // se vuelve a iniciar la recepción de datos
    }
```

Ilustración 13. Captura 3 - Captura de paquetes

```
1 referencia
protected void OnNewPacket(Paquete p)
{
    if (newPacketEventHandler != null)
    {
        newPacketEventHandler(this, p);
    }
}
```

Ilustración 14. Captura 4 - Captura de paquetes

4.2 Modo promiscuo

Para escuchar el tráfico, será necesario configurar nuestra tarjeta de red en "modo promiscuo", este modo permite interceptar todo el tráfico de red, incluso paquetes de los que no somos destinatarios.

Este modo está muy relacionado con los sniffers, que se basan en este modo para realizar su tarea. Multitud de protocolos de red no encriptan los datos, por lo tanto, es posible ver en claro datos de protocolos como, Telnet, POP, FTP...

En esta aplicación he utilizado el siguiente código para conseguir poner la tarjeta en promiscuo:

En la clase Monitor, en la función Empezar. En primer lugar, comprobamos si la dirección IP es IPv4 o IPv6, a continuación abrimos el socket y establecemos el modo operativo a bajo nivel.

```
public void Empezar()
{
    if (monitor_Socket == null)
    {
        try
        {

            if (ipAddress.AddressFamily == AddressFamily.InterNetwork)
            { // IPv4
                monitor_Socket = new Socket(AddressFamily.InterNetwork, SocketType.Raw, System.Net.Sockets.ProtocolType.IP)
            }
            else
            { // IPv6
                monitor_Socket = new Socket(AddressFamily.InterNetworkV6, SocketType.Raw, System.Net.Sockets.ProtocolType.I
            }
            // abre el socket
            monitor_Socket.Bind(new IPEndPoint(ipAddress, 0));
            // establece los modos operativos a bajo nivel para los sistemas operativos
            monitor_Socket.IOControl(SIO_RCVALL, BitConverter.GetBytes((int)1), null);
            //  se indica donde se va a realizar la recepción
            monitor_Socket.BeginReceive(buffer, 0, buffer.Length, SocketFlags.None, new AsyncCallback(this.OnReceive), null
        }
        catch (Exception e)
        {
            monitor_Socket.Close();
            monitor_Socket = null;
            Console.WriteLine(e.ToString());
        }
    }
}
```

Ilustración 15. Captura - Modo Promiscuo

4.3 Ejecutar con permisos de administrador

Uno de los problemas que me encontré, era que al ejecutar la aplicación, cuando pulsaba sobre empezar, no me capturaba ningún paquete. Esto era debido, a que la aplicación hay que ejecutarla como administrador. Buscando por Internet, encontré varios ejemplos, con la función IsUserAnAdmin(), pero no lograba que funcionará. Al final el problema, era que a partir del Windows Vista no es compatible.

Una posible solución la encontré en la siguiente página web: http://lluisvera.com/net-c-ejecutar-aplicaciones-como-administrador/

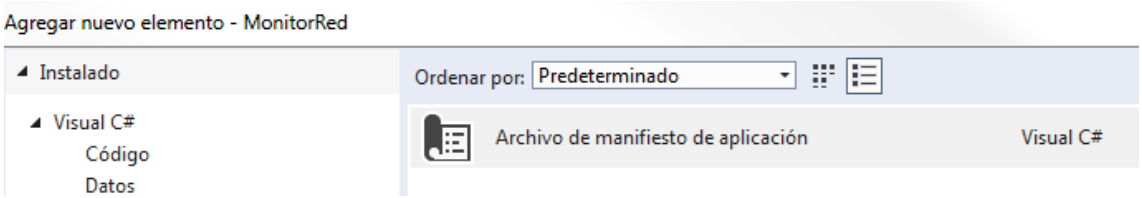


Ilustración 16. Captura 1 - Ejecutar como administrador

En primer lugar, agrego el elemento: “Archivo de manifiesto de aplicación”

Una vez tengamos nuestro fichero, hemos de hacer una pequeña modificación en el requested Execution Level para dejarlo en “**requireAdministrador**” en lugar de “**asInvoker**“.

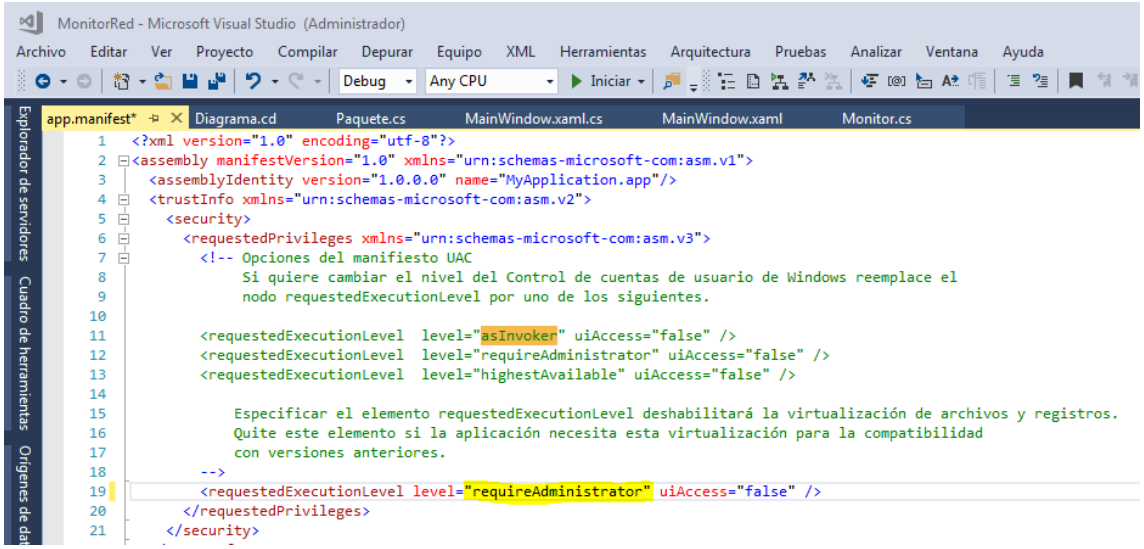


Ilustración 17. Captura 2 - Ejecutar como administrador

Por último, en las propiedades de la aplicación, elegimos el manifiesto modificado, para ejecutarse en modo administrador.

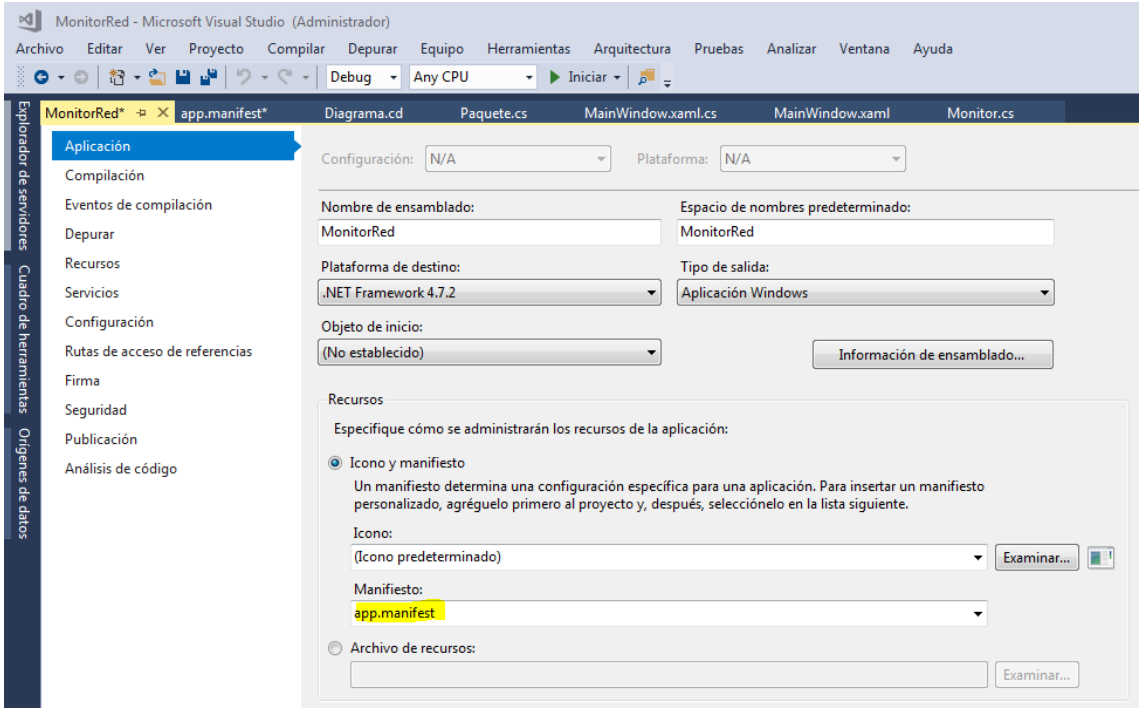


Ilustración 18. Captura 3 - Ejecutar como administrador

En la opción de aplicación, en recursos debemos especificar el icono por defecto y el fichero Manifiesto (app.manifest).

4.4 Filtrado

El filtrado se ha configurado para que pueda filtrarse por IP, Puerto o protocolo. Se pueden ampliar las opciones de filtrado, así como perfeccionar las actuales.

Como se puede ver en la captura siguiente, se optó por añadir casillas de verificación, para habilitar o deshabilitar las diferentes opciones de filtrado.

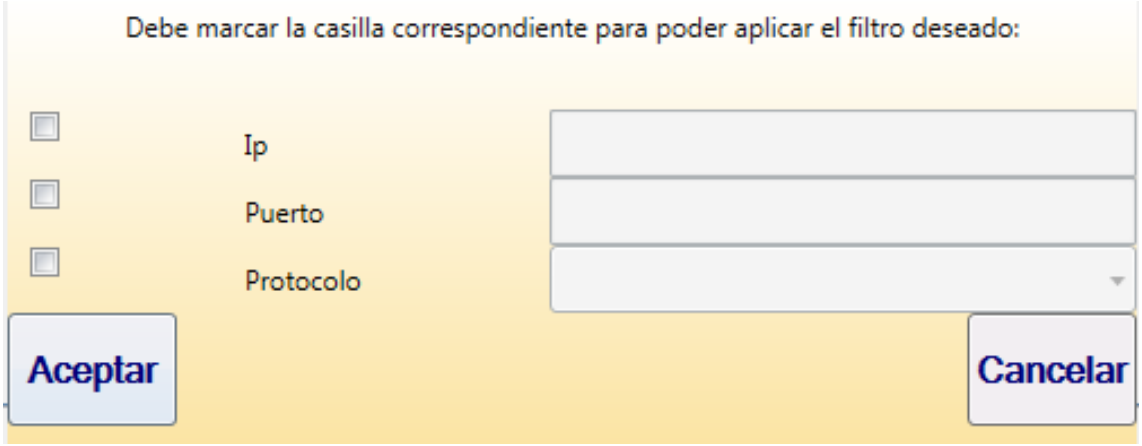
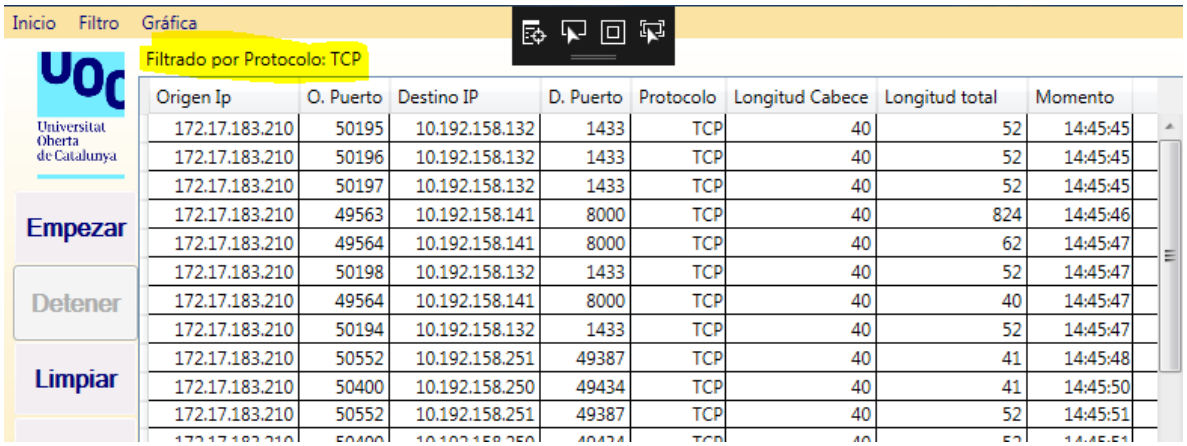


Ilustración 19. Captura 1 - Filtrado

Cuando pulsamos sobre “Aceptar”, se aplica el filtrado elegido, y en la parte superior de la pantalla principal, aparece el filtrado elegido para poder visualizarlo en la pantalla principal. En la siguiente captura, observamos que se ha filtrado por protocolos TCP, y todos los paquetes que aparecen pertenecen a este protocolo.



Origen Ip	O. Puerto	Destino IP	D. Puerto	Protocolo	Longitud Cabece	Longitud total	Momento
172.17.183.210	50195	10.192.158.132	1433	TCP	40	52	14:45:45
172.17.183.210	50196	10.192.158.132	1433	TCP	40	52	14:45:45
172.17.183.210	50197	10.192.158.132	1433	TCP	40	52	14:45:45
172.17.183.210	49563	10.192.158.141	8000	TCP	40	824	14:45:46
172.17.183.210	49564	10.192.158.141	8000	TCP	40	62	14:45:47
172.17.183.210	50198	10.192.158.132	1433	TCP	40	52	14:45:47
172.17.183.210	49564	10.192.158.141	8000	TCP	40	40	14:45:47
172.17.183.210	50194	10.192.158.132	1433	TCP	40	52	14:45:47
172.17.183.210	50552	10.192.158.251	49387	TCP	40	41	14:45:48
172.17.183.210	50400	10.192.158.250	49434	TCP	40	41	14:45:50
172.17.183.210	50552	10.192.158.251	49387	TCP	40	52	14:45:51

Ilustración 20. Captura 2 - Filtrado

Se ha implementado como un control de usuario, para facilitar posibles evoluciones de las opciones de filtrador y encapsulamiento de la funcionalidad.

Podemos encontrar la impementación en el namesapce MonitorRed.Red.Filtro, en este espacio encontraremos la clase Filtro con todos las posibilidades del filtro, y el propio control donde se observa la parte gráfica y codigo que aporta la funcionalidad al control. Cabe destacar que se han implementado dos hadlers, uno para el botón

aceptar y otro para el botón cancelar que permiten disparar sendos eventos en la pantalla principal.

En la siguiente captura, podemos ver la implementación al pulsar sobre el botón aceptar, en la ventana de filtro. La función btnAceptar_Click de la clase uscFiltro:

```
private void btnAceptar_Click(object sender, RoutedEventArgs e)
{
    this.Cursor = System.Windows.Input.Cursors.Wait;
    try
    {
        FiltroAceptarEventArgs args = new FiltroAceptarEventArgs();
        Filtro f = new Filtro();

        // si han seleccionado el check correspondiete
        /*************************/

        // validar ip

        f.FiltroIP = null;
        f.FiltroPuerto = null;
        f.FiltroProtocolo = null;
        if (chkIp.IsChecked == true &&  txtIp.Text != "")
        {
            Regex regexpr = new Regex(@"^\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}$");
            Match match = regexpr.Match(txtIp.Text.Replace("_",""));
            if (match.Success)
            {
                f.FiltroIP = txtIp.Text;

            } else
            {
                MessageBox.Show("El formato de la dirección ip debe ser ###.###.###.###");
                return;
            }

        }

        if (chkPuerto.IsChecked == true && txtPuerto.Text != "")
        {
            int auxval;
            bool auxNumber = int.TryParse(txtPuerto.Text, out auxval);
            if (auxNumber == true)
            {
```

Ilustración 21. Captura 3 - Filtrado

4.5 Gráfica

Se ha optado por realizar una gráfica, para poder sacar estadísticas, para poder tener una visión más clara de los paquetes capturados y visualizar la cantidad de paquetes según el protocolo capturado.

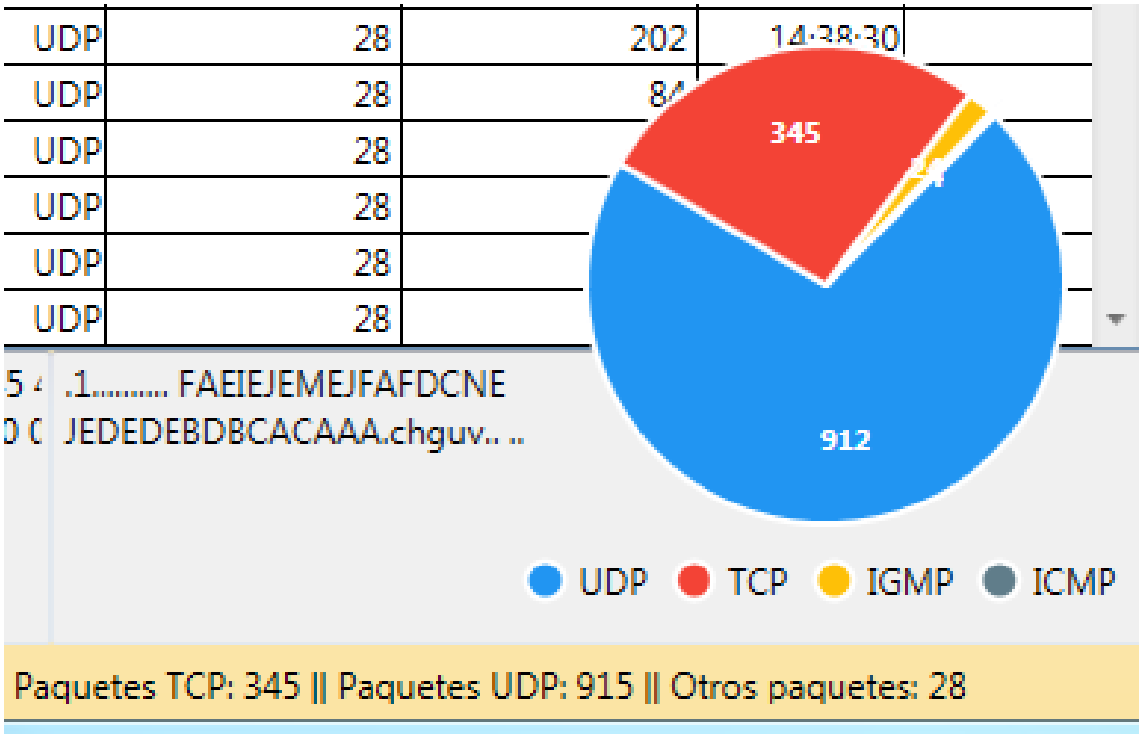


Ilustración 22. Captura 1 - Gráfica

La forma elegida ha sido un gráfico circular, este tipo de gráfico permite presentar, de forma gráfica, ordenada y clasificada, información estadística. De esta forma, puede ser posteriormente analizada en busca de problemas de red. Ayudan a observar con mayor claridad la evolución de los datos y facilitan la comparación de los mismos en diferentes espacios de tiempo.

Para realizar la gráfica he utilizado la biblioteca gratuita LiveCharts (https://lvcharts.net/), una librería Open Source para WPF que permite realizar gráficas, cambios sobre las mismas y actualizaciones en tiempo real.

Se utilizaron las referencias de Nugets, para ver cómo se utilizaron ver el punto 10.2.

Para averiguar el funcionamiento y su configuración, me base en el ejemplo de la siguiente URL:

https://ourcodeworld.com/articles/read/583/how-to-create-a-pie-chart-using-the-livecharts-library-in-winforms-c-sharp

Uno de los cambios realizados, es asignarle a los valores de la gráfica, las cantidades de paquetes por protocolo. Si se muestra la gráfica durante el proceso de captura de

tráfico, la gráfica circular se va actualizando a medida que se van capturando los paquetes.

Para la configuración de la gráfica, he implementado la función Grafico2, de la clase MainWindow:

```
private void Grafico2()
{
    if (grdgraficoChart.Visibility != Visibility.Visible)
    {
        return;
    }

    PointLabel = chartPoint =>
        string.Format("{0} ({1:P})", chartPoint.Y, chartPoint.Participation); //Formato del label del gráfico

    var chart = (LiveCharts.Wpf.PieChart)graficoChart; //inicia una instancia de Tipo de gráfico circular

    chart.Series.Clear();
    PieSeries r = null;

    List<Paquete> lst = new List<Paquete>(this.paquetesLeidos); //crear lista con los paquetes capturados
    List<String> lstx = lst.Select(p => p.Type).Distinct().ToList(); //crear una lista con los distintos tipos de protocolos
    int auxNum = 0;

    foreach (String pro in lstx)
    {
        auxNum = 0;
        auxNum = lst.Where(c => c.Type == pro).Count(); //contador paquetes por protocolo

        r = new PieSeries
        {
            //Caracteristicas que aparecen en la gráfica
            Title = pro,
            Values = new ChartValues<double> { auxNum },
            DataLabels = true
        };

        chart.Series.Add(r); //añade la serie
    }
```

Ilustración 23. Captura 2 - Gráfica

4.6 Guardar

Para la opción “Guardar”, en un principio conseguí guardar los paquetes capturados, directamente en C:\ en un fichero llamado “PaquetesGuardados.txt”.

```
private void mnuInicioGua_Click(object sender, EventArgs e)
        {

            StreamWriter ContenidoGuardado = File.CreateText("c:\\PaquetesGuradados.txt");
            String contenido = grdDatos.ItemsSource.ToString();
            ContenidoGuardado.Write(contenido.ToString());
            ContenidoGuardado.Flush();
            ContenidoGuardado.Close();
            MessageBox.Show("Paquetes guardados en C:\\PaquetesGuardados.txt");
        }
```

Luego opte por buscar una solución, para que al pulsar guardar nos de la opción de elegir donde queremos guardar y el nombre que queremos darle, para evitar no saber dónde se guarda el fichero.

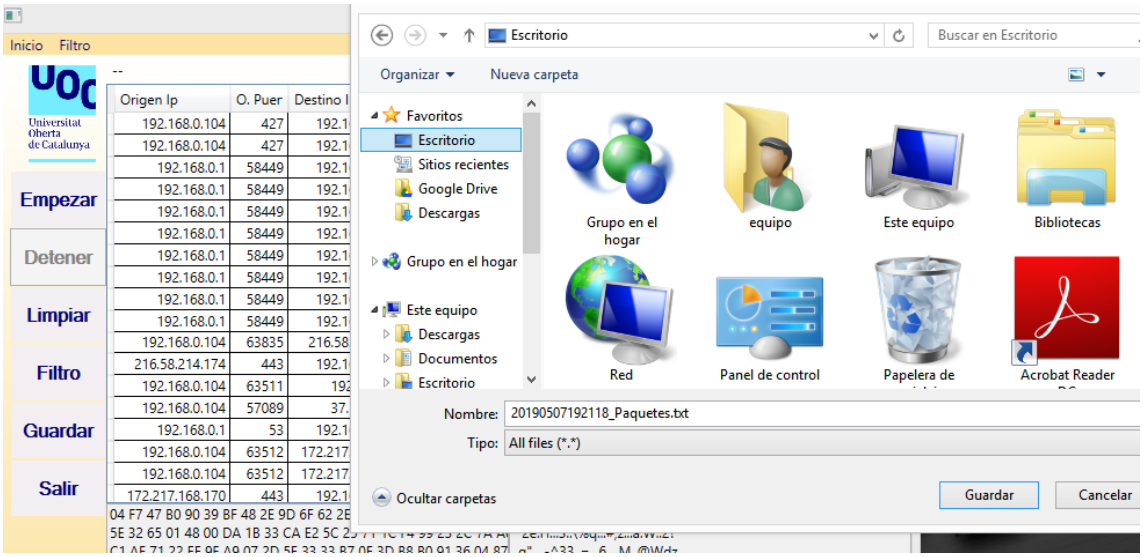


Ilustración 24. Captura 1 - Guardar

Para ello utilice el componente SaveFileDialog, este componente nos muestra la ventana para seleccionar la ubicación donde se guardará el archivo. Me he guiado de la información obtenida en la siguiente URL:
https://www.c-sharpcorner.com/uploadfile/mahesh/savefiledialog-in-C-Sharp/

```
private void mnuInicioGua_Click(object sender, EventArgs e)
{

    SaveFileDialog saveFileDialog1 = new SaveFileDialog();

    saveFileDialog1.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*"; //Solo archivos txt
    saveFileDialog1.FilterIndex = 2; //índice del filtro seleccionado
    saveFileDialog1.RestoreDirectory = true; //Restaura el directorio actual antes de cerrarse.
    saveFileDialog1.FileName = DateTime.Now.ToString("yyyyMMddHHmmss") + "_Paquetes.txt"; //Nombre

    if (saveFileDialog1.ShowDialog() != true)
    {
        return;
    }

    StreamWriter ContenidoGuardado = File.CreateText(saveFileDialog1.FileName);
    List<Paquete> contenido = (List<Paquete>)grdDatos.ItemsSource;

    //Formato documento guardado
    foreach (Paquete lin in contenido)
    {
        string s = lin.Des_IP + "|";
        s += lin.Des_PORT + "|";
        s+= lin.Src_IP + "|";
        s += lin.Src_PORT + "|";
        s += lin.Time + "|";
        s += lin.Type + "|";
        s += lin.HeadLength + "|";
        s += lin.TotalLength + "|" + System.Environment.NewLine;
        s += lin.getHexString() + "|" + System.Environment.NewLine;
         s += lin.getCharString() + System.Environment.NewLine + "----------------------------";
        ContenidoGuardado.WriteLine(s);
    }

    ContenidoGuardado.Flush();
    ContenidoGuardado.Close();
    MessageBox.Show("Paquetes guardados en " + saveFileDialog1.FileName);
}
```

Ilustración 25. Captura 2 - Guardar

5. Manual de usuario

En primer lugar, procedemos a la instalación de la aplicación, mediante el instalador generado a través de Microsoft Visual Studio, con la extensión “Microsoft Installer”. Podemos ver los pasos de creación del instalador e instalación de la aplicación en el anexo 10.3.

Una vez instalado la aplicación, vamos a explicar los pasos para la utilización de la aplicación:

Se creará un acceso directo en el escritorio.

Una vez ejecutamos el acceso directo, nos aparecerá la ventana principal de la aplicación.

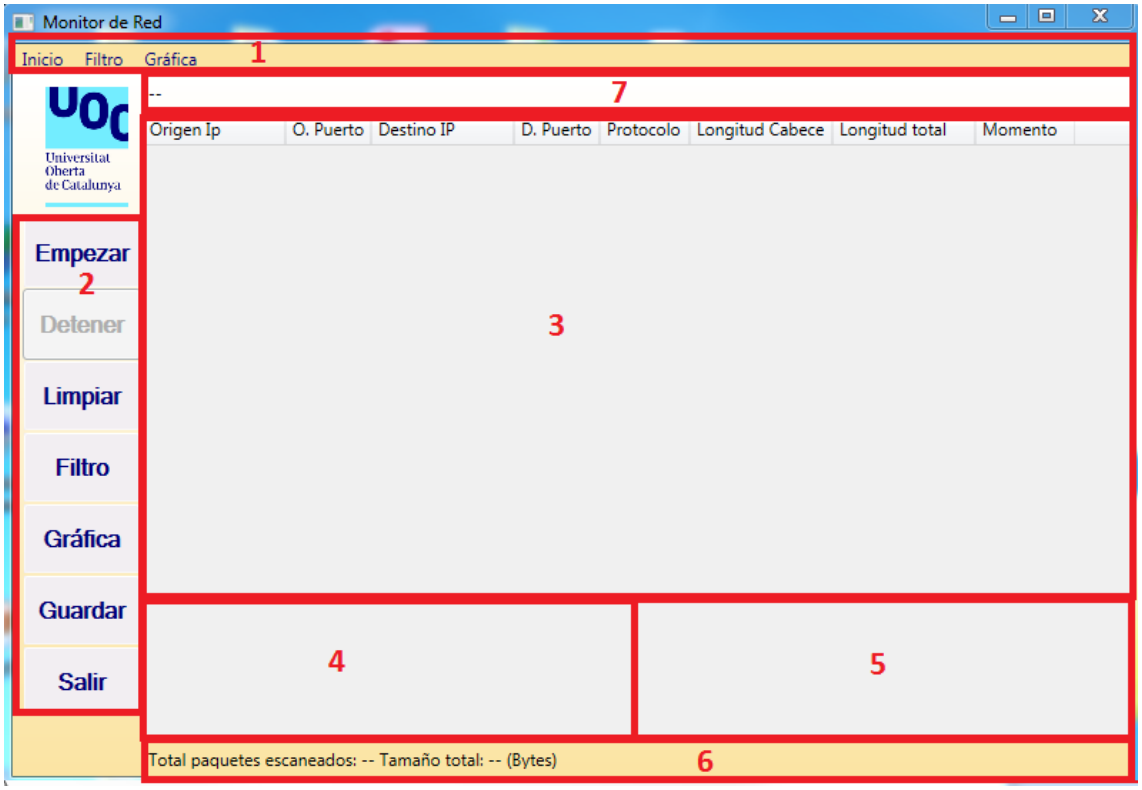


Ilustración 26. Captura 1 - Manual de usuario

A continuación, señalo las diferentes partes de la aplicación, marcadas con un número en la captura anterior:

1. Panel menú donde aparecen todas las opciones del programa.
2. Panel con opciones principales de la aplicación.
3. Espacio donde aparecerán los paquetes capturados.
4. Espacio donde aparecerá el paquete seleccionado de la lista, en formato hexadecimal.
5. Espacio donde aparecerá el paquete seleccionado de la lista, en formato caracter.
6. Espacio donde aparecerán el total de paquetes escaneados, tamaño total, total paquetes TCP, UDP y otros protocolos.
7. Espacio donde aparecen los filtros seleccionados.

Para comenzar a capturar los paquetes, debemos pulsar en el botón “Empezar”, a continuación, se nos abrirá una nueva ventana, para que elijamos el dispositivo de red, por el cual queremos empezar a capturar los paquetes:

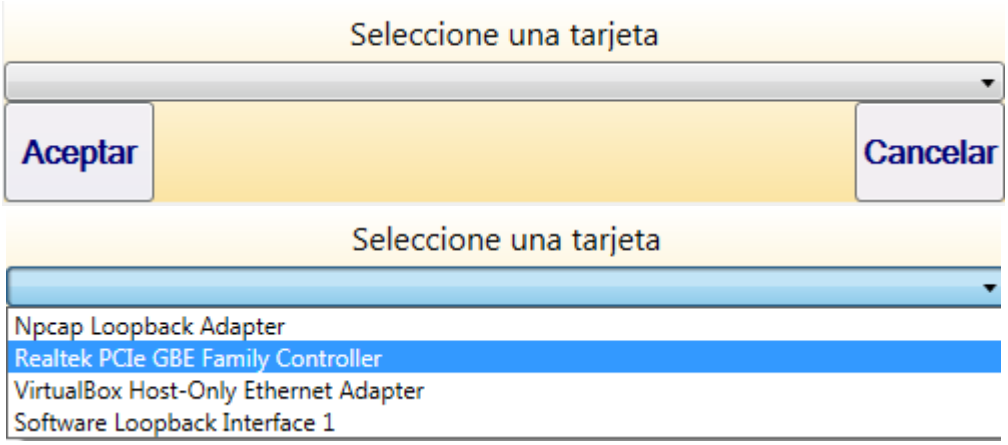


Ilustración 27. Captura 2 - Manual de usuario

Una vez seleccionada el dispositivo, pulsamos en aceptar, y la aplicación empieza a mostrar los paquetes capturados:

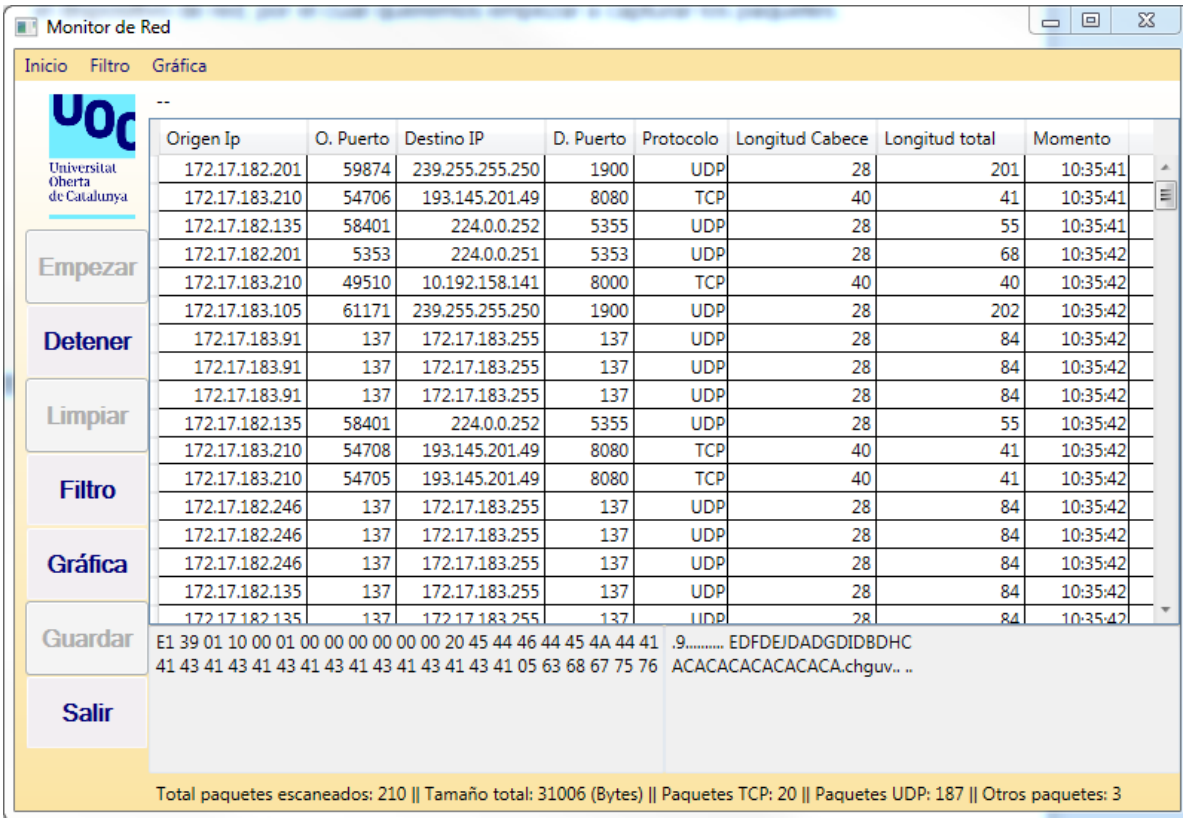


Ilustración 28. Captura 3 - Manual de usuario

Como podemos observar, en la parte 7, aparecen los totales actuales de los paquetes escaneados, tamaño total, paquetes TCP, UDP y Otros protocolos. También podemos ver en la parte 4 y 5, en formato hexadecimal y carácter respectivamente, el último paquete donde teníamos el cursor.

Si pulsamos sobre el botón “Gráfica”, mostraremos u ocultaremos la gráfica circular, con los totales de paquetes por protocolos.

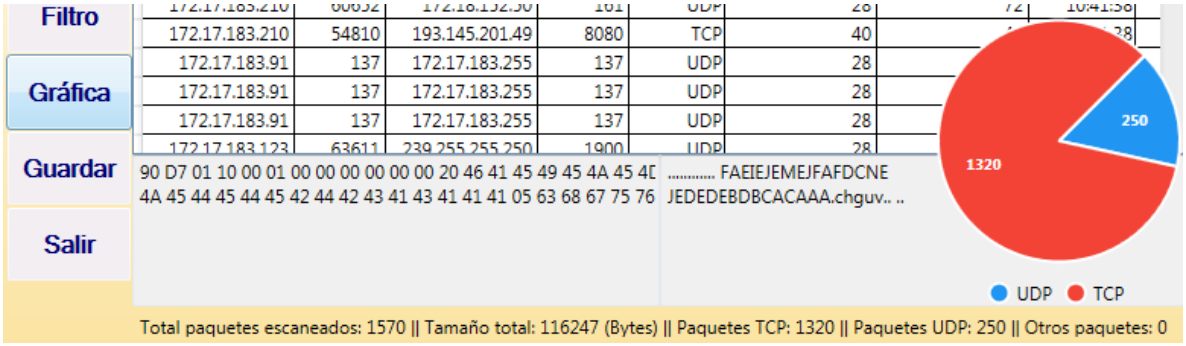


Ilustración 29. Captura 4 - Manual de usuario

Con el botón “Limpiar” vaciaremos todos los paquetes capturados, así como todos los contadores y filtros seleccionados.

Con el botón “Filtrar”, podemos filtrar la busqueda de paquetes, por IP, Puerto o Protocolo:

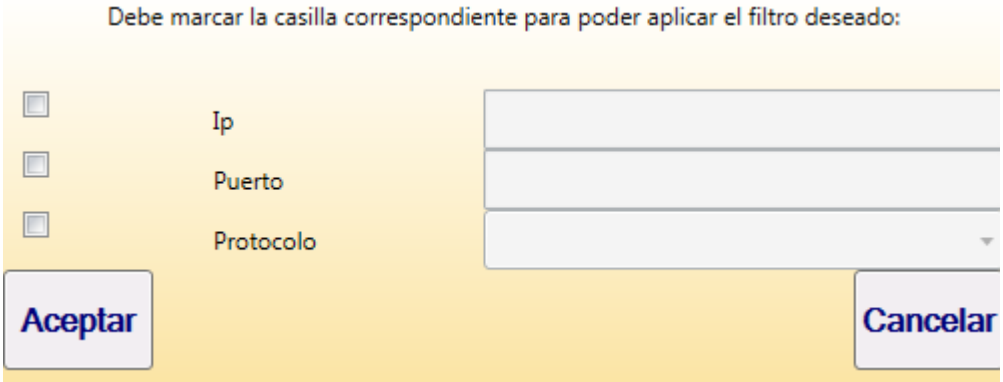


Ilustración 30. Captura 5 - Manual de usuario

Por defecto estan las opciones deshabilitadas, para habilitarlas debemos marcar el checkbox sobre la opción que deseamos. A continuación, debes escribir o seleccionar el valor del filtro deseado.

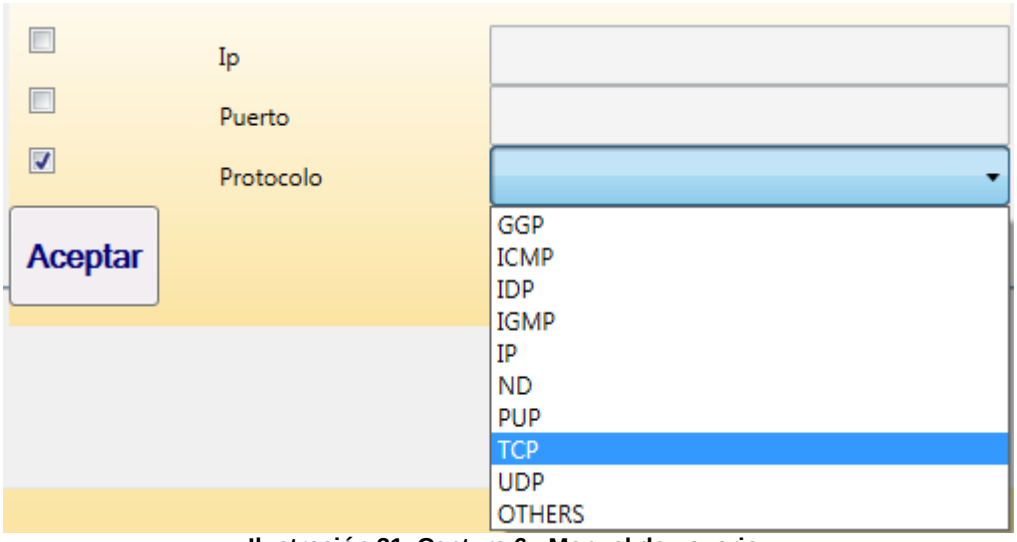
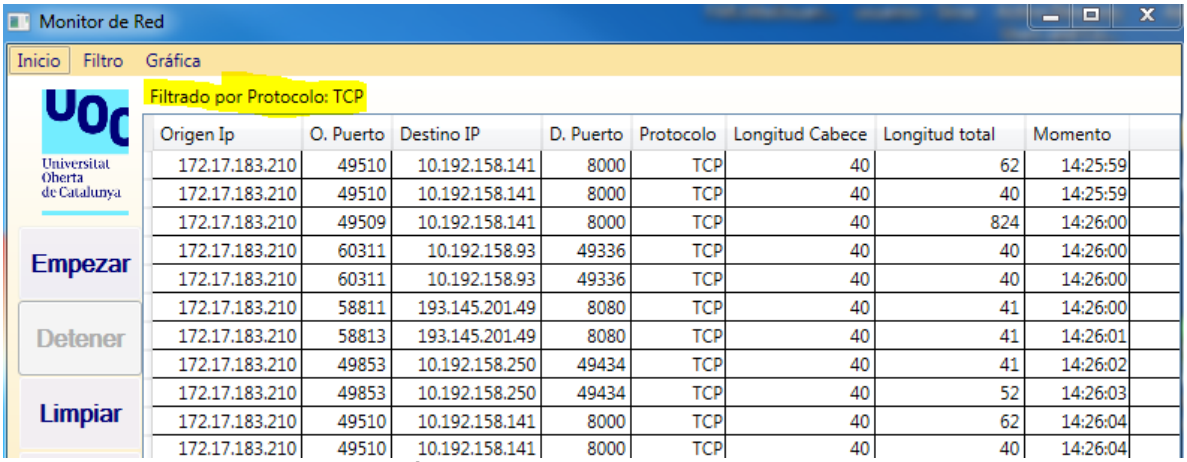


Ilustración 31. Captura 6 - Manual de usuario

Por último, si volvemos a pulsar sobre “Empezar”, la nueva busqueda será solo por el filtro seleccionado, en la siguiente captura se observa, como solo se estan capturando paquetes del protocolo TCP.



Origen Ip	O. Puerto	Destino IP	D. Puerto	Protocolo	Longitud Cabece	Longitud total	Momento
172.17.183.210	49510	10.192.158.141	8000	TCP	40	62	14:25:59
172.17.183.210	49510	10.192.158.141	8000	TCP	40	40	14:25:59
172.17.183.210	49509	10.192.158.141	8000	TCP	40	824	14:26:00
172.17.183.210	60311	10.192.158.93	49336	TCP	40	40	14:26:00
172.17.183.210	60311	10.192.158.93	49336	TCP	40	40	14:26:00
172.17.183.210	58811	193.145.201.49	8080	TCP	40	41	14:26:00
172.17.183.210	58813	193.145.201.49	8080	TCP	40	41	14:26:01
172.17.183.210	49853	10.192.158.250	49434	TCP	40	41	14:26:02
172.17.183.210	49853	10.192.158.250	49434	TCP	40	52	14:26:03
172.17.183.210	49510	10.192.158.141	8000	TCP	40	62	14:26:04
172.17.183.210	49510	10.192.158.141	8000	TCP	40	40	14:26:04

Ilustración 32. Captura 7 - Manual de usuario

Si queremos guardar todos los paquetes escaneados, tenemos la opción de “Guardar”, al pulsar esta opción, se nos abrirá una ventana donde seleccionaremos el lugar donde queremos guardar el fichero y el nombre que queremos ponerle, por defecto, nos pondrá la fecha y hora actual.

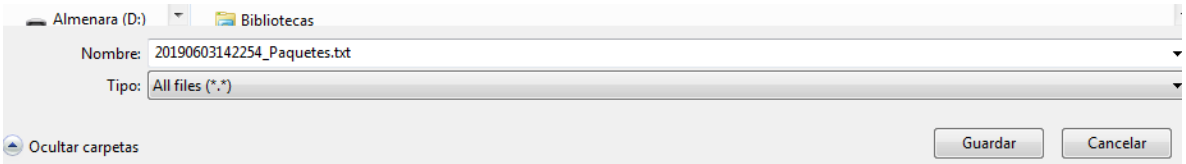


Ilustración 33. Captura 8 - Manual de usuario

El fichero que crea, lo podemos ver en la siguiente captura. Está configurado para que separe cada paquete por líneas, y muestre todas las opciones del paquete en cuestión, junto a su formato hexadecimal y carácter.

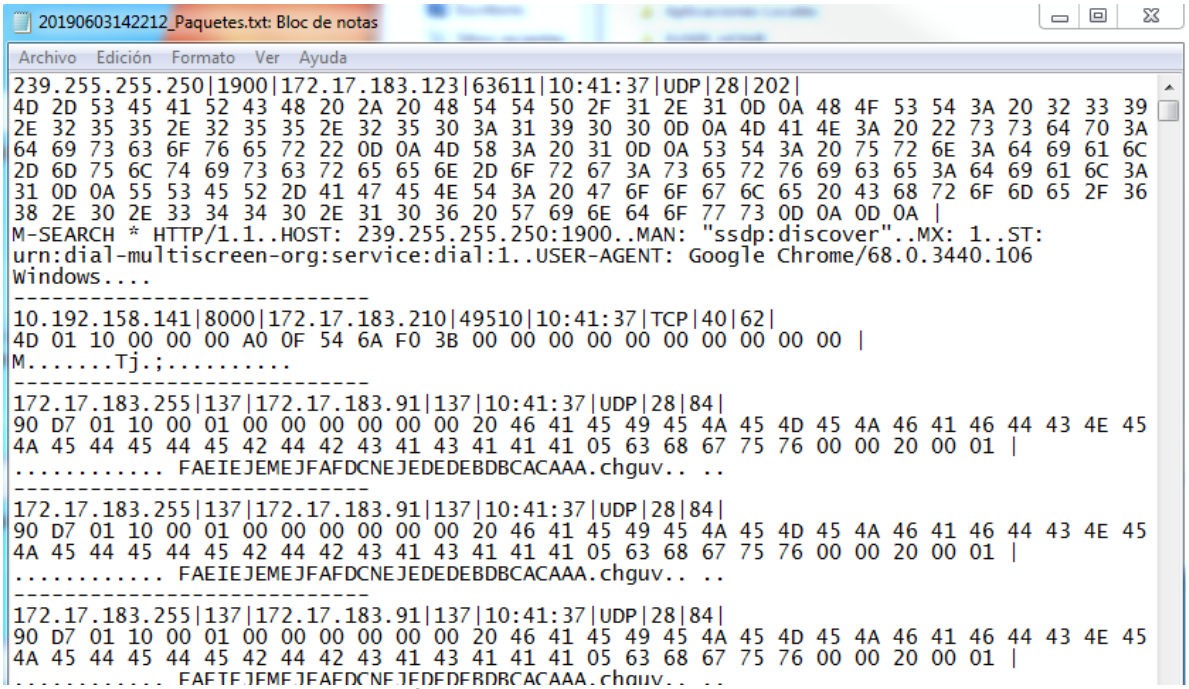


Ilustración 34. Captura 9 - Manual de usuario

6. Caso Práctico

Durante una semana se estuvo utilizando la aplicación en mi equipo del hospital, para comprobar el funcionamiento correcto de la aplicación y para analizar los diferentes paquetes que pasan por el dispositivo de red, en busca de posibles malas configuraciones o fallos de red, así como, detectar fallos de seguridad o búsqueda de datos no encriptados que circulen por la red.

A continuación, muestro capturas de diferentes días en distintas horas, en ellas podemos observar la cantidad de paquetes que circulan por la red, y como el programa los va mostrando gráficamente y estadísticamente.

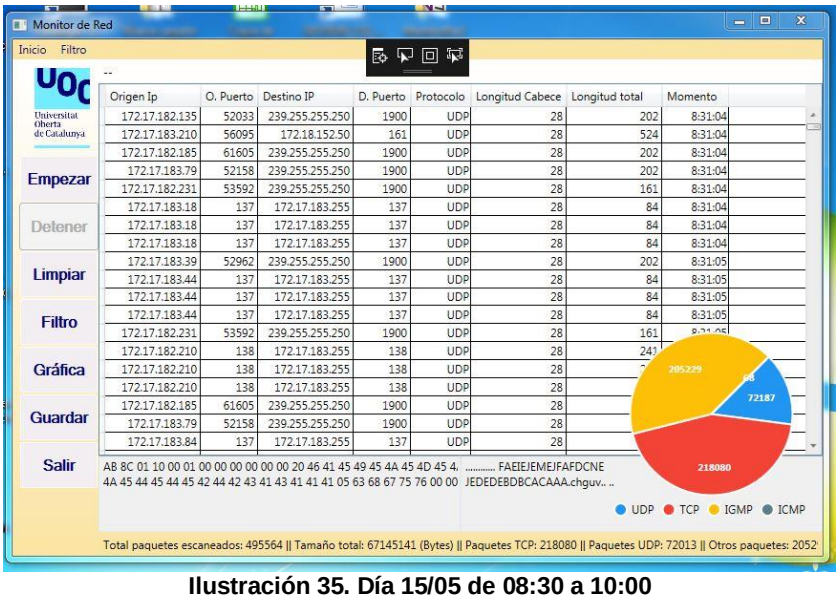


Ilustración 35. Día 15/05 de 08:30 a 10:00

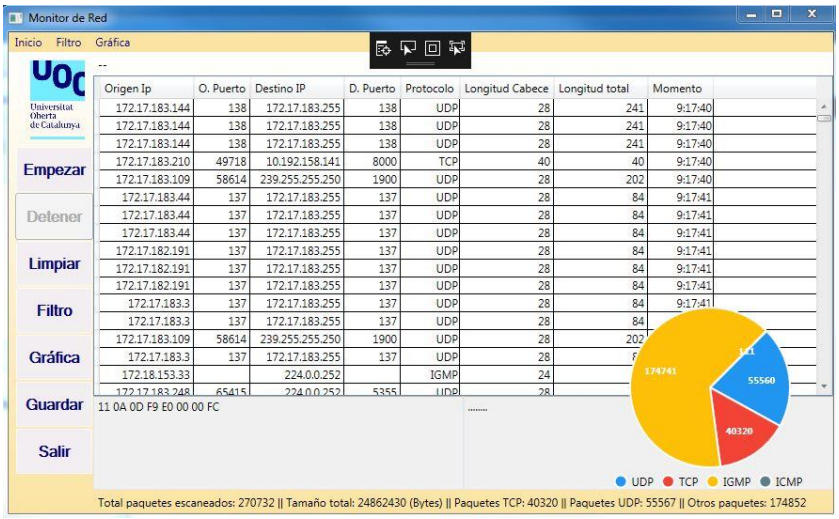


Ilustración 36. Día 16/05 de 09:17 a 10:36

Los archivos guardados en los diferentes días, permiten hacer una búsqueda más exhaustiva en busca de posibles fallos de red o de seguridad. Se ha detectado que en los días que ha estado durante más tiempo escaneando, el fichero ha crecido demasiado, lo que dificulta trabajar con el mismo, debido al tiempo que necesita el equipo para abrirlo.

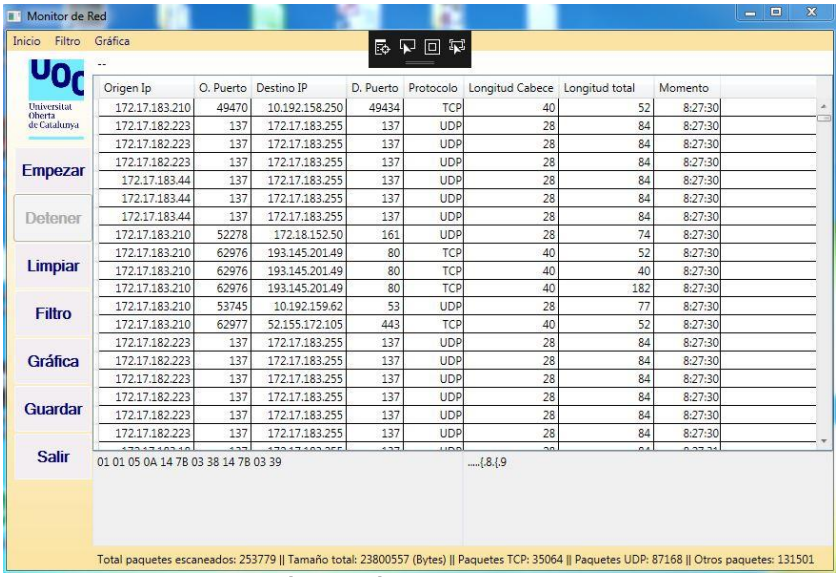


Ilustración 37. Día 20/05 de 08:27 a 09:47

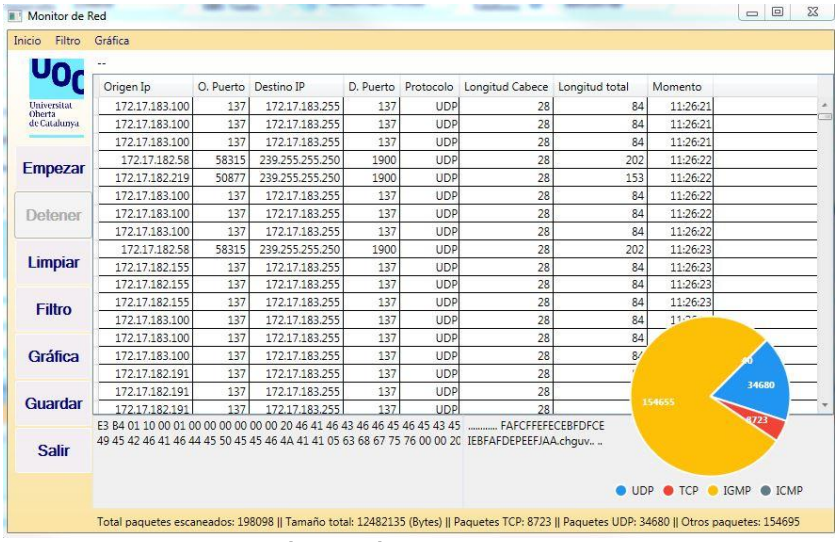


Ilustración 38. Día 22/05 de 11:26 a 13:00

Se observa que los paquetes del protocolo IGMP, son los más capturados durante todos los días que se ha utilizado la aplicación. IGMP especifica de qué manera un host puede registrarse con un router para recibir tráfico multicast específico.

Los mensajes multicast que recibe el host del mismo grupo en dos subredes distintas pueden ser distintos también. Por ejemplo, 244.0.0.1 es el grupo para "todos los hosts de esta subred", por lo que los mensajes recibidos en una subred siempre serán diferentes para este grupo que para los de otra subred. Puede haber múltiples procesos en un host a la escucha de mensajes para un grupo de multicast.

En la siguiente captura, donde se ha aplicado un filtro por protocolo IGMP, se puede observar los mensajes desde diferentes equipos al router utilizando direcciones multicast 244.0.0.1.

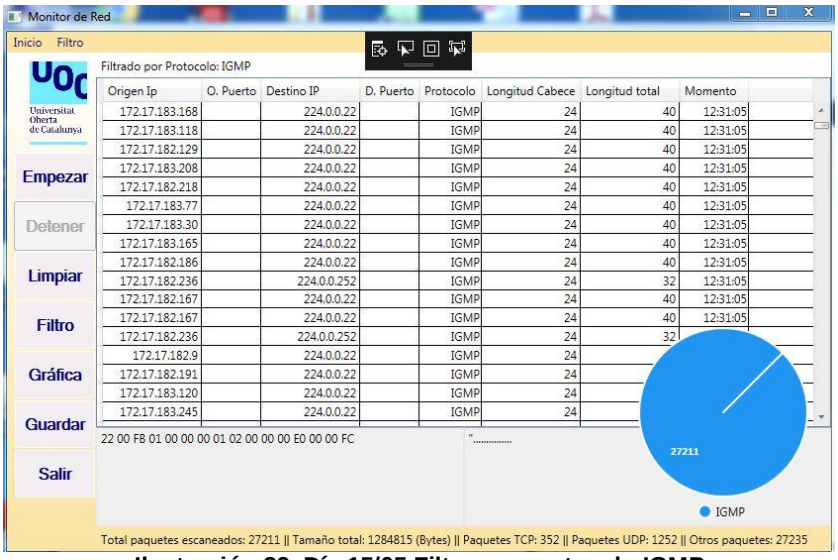


Ilustración 39. Día 15/05 Filtro por protocolo IGMP

En las dos siguientes capturas se observa un problema con la seguridad, relacionada con el correo corporativo del hospital, este funciona con el protocolo SMTP Y POP3. De hecho, cuando enviamos un correo utilizando el protocolo SMTP o lo recibimos a través de POP3, tenemos que saber que tanto el usuario como la contraseña de la cuenta se envían en texto plano sin encriptar. Es decir, cualquier persona conectada a nuestra red o a la del servidor con un sniffer de red (capturador de paquetes) debidamente configurado podría ver esa información y, por supuesto, interceptar los paquetes dentro de los que viajan los datos de nuestros correos electrónicos.

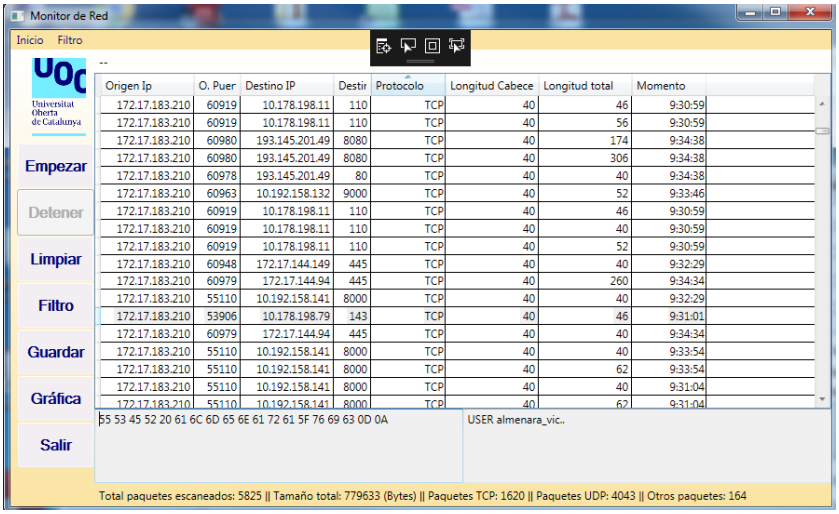


Ilustración 40. Detección problema de seguridad correo

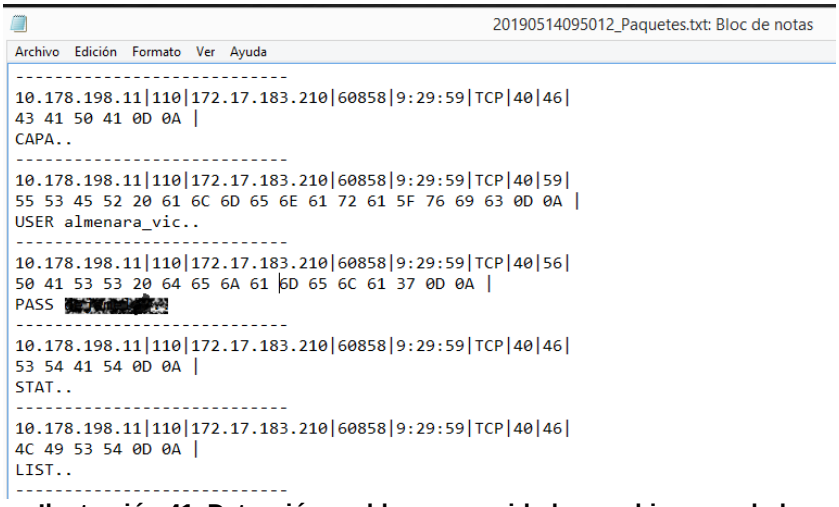


Ilustración 41. Detección problema seguridad en archivo guardado

Se procede a buscar información sobre como encriptar toda la información del correo, obteniendo resultados para configurar programas como Microsoft Outlook o Thunderbird, ampliamente utilizados en los equipos del hospital.

Para Microsoft Outlook, iremos a Archivo/Información/Configuración de la cuenta y pincharemos en la opción Configuración de la cuenta. A continuación, aparecerá la ventana de configuración de las cuentas de correo electrónico, donde no tendremos más que hacer doble clic sobre la dirección cuya configuración queremos modificar. En la nueva ventana que aparezca pincharemos en Más configuraciones e iremos a la pestaña Avanzadas. Aquí, tendremos que pinchar en los desplegables correspondientes, y marcar la opción SSL o TLS.

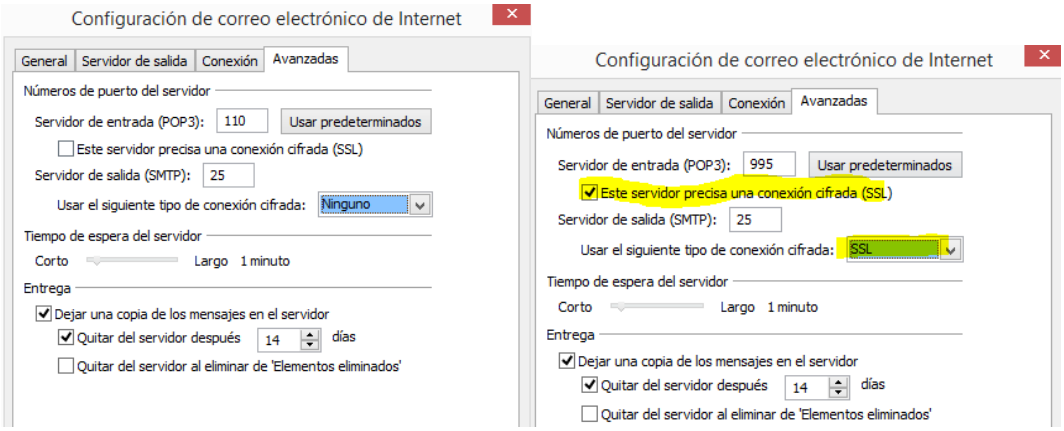


Ilustración 42. Configuración seguridad Microsoft Outlook

Para Thunderbird, se ha encontrado una extensión que añade al programa, un cifrado y autenticación de mensajes mediante OpenPGP. Ofrece cifrado y descifrado automático y una funcionalidad integrada de administración de claves. La extensión la podemos encontrar en: https://addons.thunderbird.net/es/thunderbird/addon/enigmail/

7. Conclusiones

En este trabajo de final de grado, se ha conseguido alcanzar los objetivos tanto personales como de la aplicación a desarrollar. Se ha creado una herramienta, que a pesar de sus limitaciones, permite realizar las principales funciones para capturar y analizar los paquetes de una red. Se ha aprendido a utilizar correctamente Microsoft Visual Studio y a desarrollar con el lenguaje de programación C# una aplicación desde cero, así como profundizar en el conocimiento de este lenguaje.

Una posible reflexión crítica sobre el logro de los objetivos planteados inicialmente sería que al final, se ha conseguido menos información, que en un principio se pensaba que se podría obtener. En un principio, se pensó en una herramienta de análisis de rendimiento de red, una monitorización de red, de servidores (basados en agentes o de forma remota) y de aplicaciones. Con funcionalidades completas de informes, alertas, etc. Aunque finalmente, visto las dificultades y el tiempo disponible, se optó por una herramienta que captura los paquetes de red, que posteriormente se pueden analizar en busca de problemas de red, seguridad, etc.

El seguimiento de la planificación ha tenido algunas desviaciones, debido a que, le he dedicado el mayor tiempo a programar las diferentes funcionalidades de la herramienta.

Por último, voy a señalar posibles nuevas líneas de trabajo y de mejora en la herramienta desarrollada:

- La posibilidad de programar una opción para avisar al usuario, vía correo electrónico, si durante el escaneo de paquetes, se detectan una gran cantidad de un paquete de un protocolo en concreto, o si detecta algún fallo de seguridad que nosotros le indiquemos previamente.
- Ampliar el abanico de protocolos que la aplicación detecte y muestre al usuario.
- Realizar un análisis más profundo sobre el filtrado de protocolos, se han implementado los filtros más básicos, pero con la información capturada se podrían realizar filtros más complejos. Un ejemplo, sería filtrar por tamaño del paquete.
- Añadir una función de buscar, para encontrar más fácilmente un dato de un paquete en concreto.
- Mejorar la fluidez de la aplicación, almacenando los datos en base de datos, ya que la cantidad de paquetes que almacena hace que la aplicación ralentice el sistema.
- Desarrollar la aplicación para diferentes sistemas operativos, como, por ejemplo, para aplicaciones móviles (Android).

8. Glosario

Datagrama: Paquete de datos que constituye el mínimo bloque de información en una red de conmutación por datagramas.

Unidireccional: Una relación unidireccional sólo es válida en una dirección

Bidireccional: Una relación bidireccional es válida en las dos direcciones.

Multiplexación: Combinación de dos o más canales en un solo medio de transmisión.

Monitoreo de red: Describe el uso de un sistema que constantemente monitoriza una red de computadoras en busca de componentes defectuosos o lentos.

Encriptar: Es una manera de codificar la información para protegerla frente a terceros.

Protocolo Criptográfico: Protocolo abstracto o concreto que realiza funciones relacionadas con la seguridad, aplicando métodos criptográficos.

NuGet: Gestor de paquetes gratuito y de código abierto diseñado para la plataforma de desarrollo de Microsoft.

Cliente-Servidor: modelo de diseño de software en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes.

Sistema hexadecimal: Sistema de numeración posicional que tiene como base el 16. Sus números están representados por los 10 primeros dígitos de la numeración decimal, y el intervalo que va del número 10 al 15 están representados por las letras del alfabeto de la ‘A’ a la ‘F’.

9. Bibliografía

Monitorización de redes https://es.wikipedia.org/wiki/Monitorizaci%C3%B3n_de_redes 10 de marzo del 2019

La importancia del software de monitoreo de red en una empresa http://www.computing.es/infraestructuras/noticias/1109344001801/importancia-del-software-de-monitoreo-de-red-empresa.1.html 12 de marzo del 2019.

Capa de aplicación https://es.wikipedia.org/wiki/Capa_de_aplicaci%C3%B3n 13 de marzo del 2019

¿Qué es el protocolo IP? http://redesteleco.com/que_es_protocolo_ip/ 17 de marzo del 2019

Protocolo de datagramas de usuario https://es.wikipedia.org/wiki/Protocolo_de_datagramas_de_usuario 22 de marzo del 2019

Segmento TCP https://es.wikipedia.org/wiki/Segmento_TCP 22 de marzo del 2019

José Dordoigne, Redes informáticas - 6º edición -Editorial ENI – Barcelona -febrero 2018.

10. Anexos

10.1 Primeros pasos En Microsoft Visual Studio

Se instala Microsoft Visual Studio 2017 para realizar la aplicación de monitoreo de paquetes.

Se crea el proyecto:

Archivo\ Nuevo proyecto

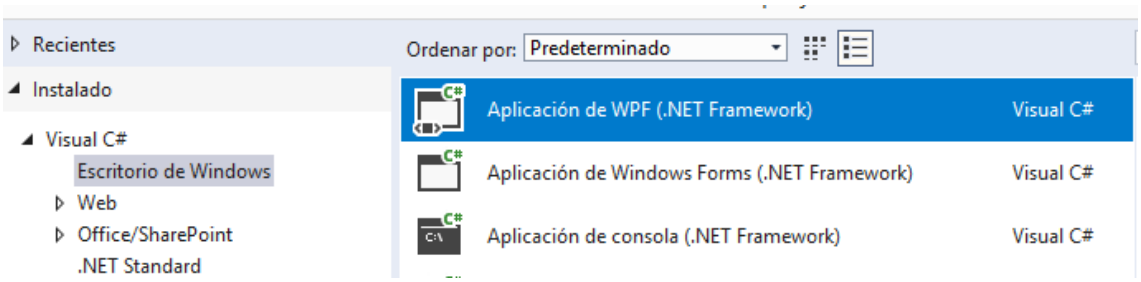


Ilustración 43. Captura 1 - Instalación Visual Studio

Lo nombramos como MonitorRed, Creamos algunos directorios para tener organizado el codigo.

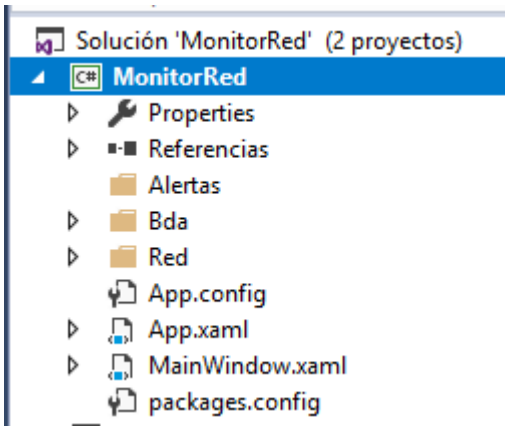


Ilustración 44. Captura 2 - Instalación Visual Studio

10.2 Añadir un paquete NuGet

Para añadir un paquete, pulsamos botón derecho sobre “Referencias”, y seleccionamos “Administrar paquetes NuGet…”

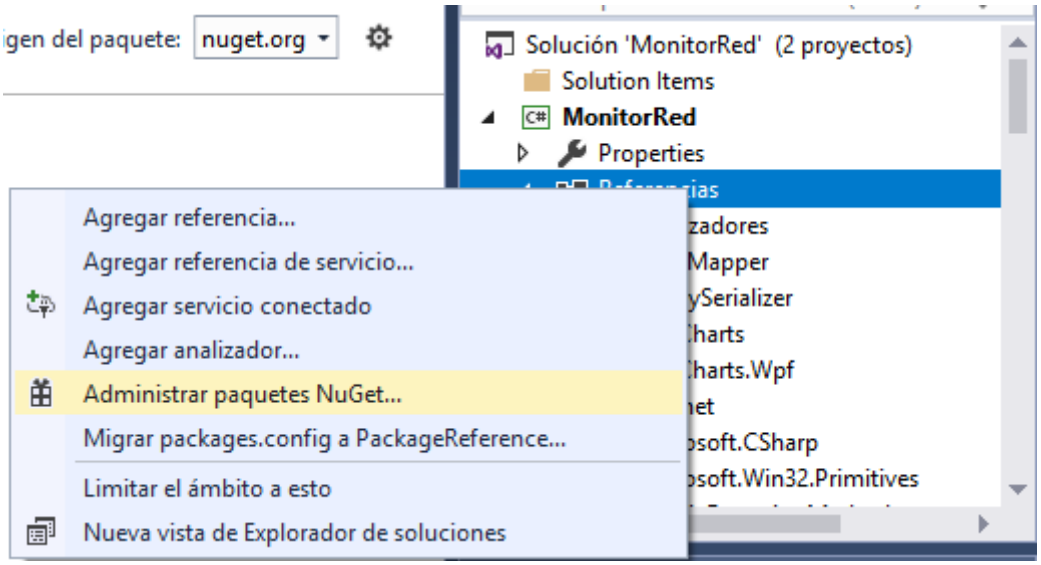


Ilustración 45. Captura 1 - Añadir NuGet

A continuación, buscamos el paquete deseado, en este ejemplo vamos a añadir, LiveCharts, que hemos utilizado para la configuración de la gráfica.

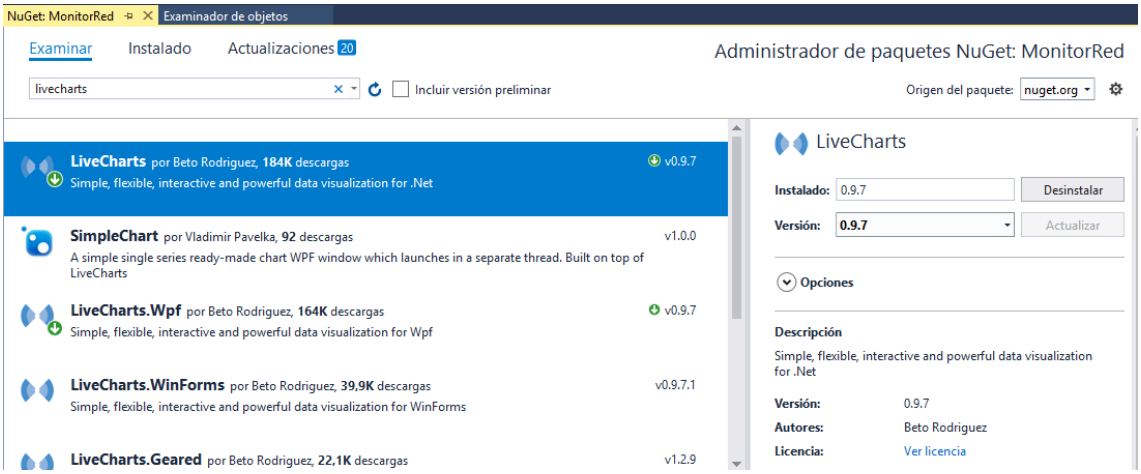
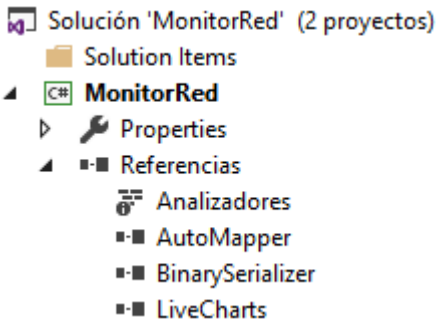


Ilustración 46. Captura 2 - Añadir NuGet

Por último le damos a instalar, una vez instalado, nos aparecerá en “Referencias”.



10.3 Crear instalador de la aplicación

En primer lugar instalamos la extensión “Microsoft Visula Studio Installer”, la podemos descargar de https://marketplace.visualstudio.com.

Una vez instalada, hacemos botón derecho sobre nuestra solución, y agregamos un nuevo proyecto. En nuestro caso lo nombramos como “SetupMonitorRed”.

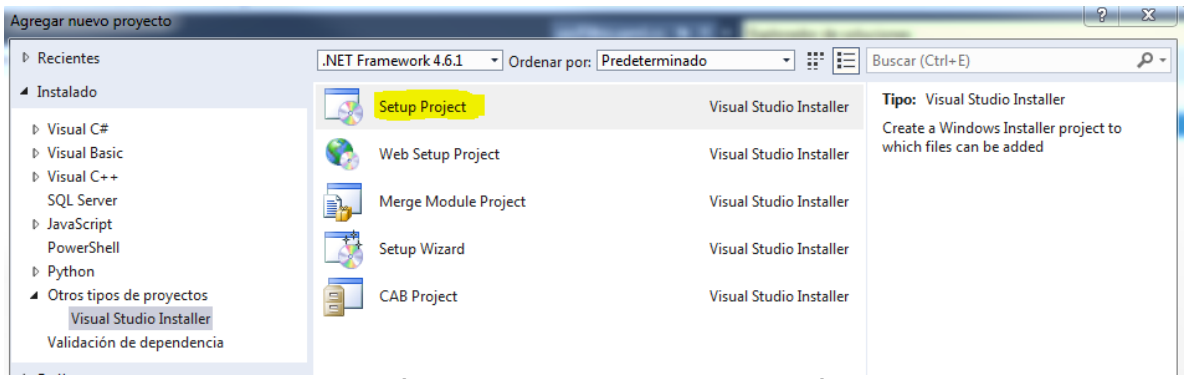


Ilustración 47. Captura 1 - Instalador aplicación

A continuación se nos abre el sistema de archivos del proyecto creado.

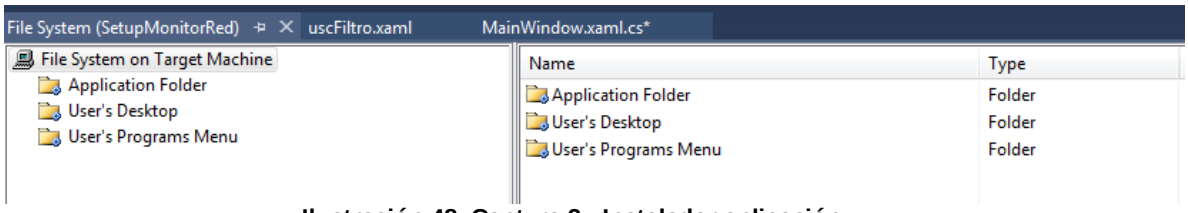


Ilustración 48. Captura 2 - Instalador aplicación

Para empezar a configurar el instalador, pulsamos botón derecho sobre “Application Folder”, Add/Resultados del proyecto…

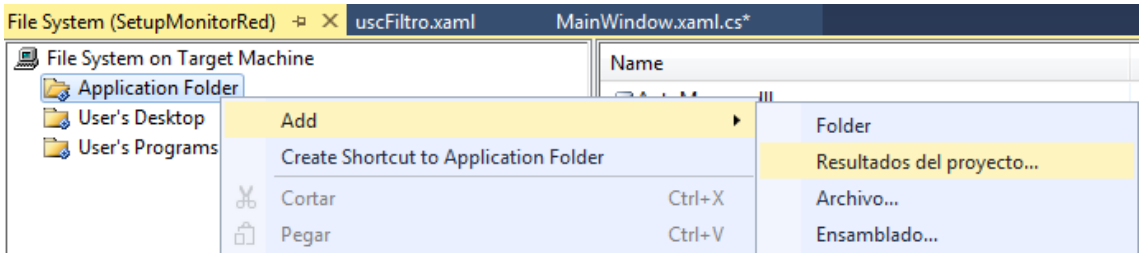
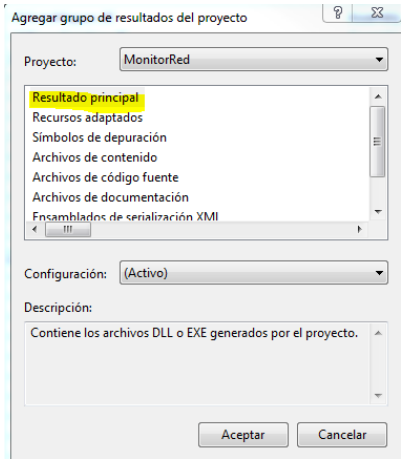


Ilustración 49. Captura 3 - Instalador aplicación



Marcamos “Resultado Principal” y aceptamos. Una vez terminado saldrán todas las dependencias que tenga el proyecto.

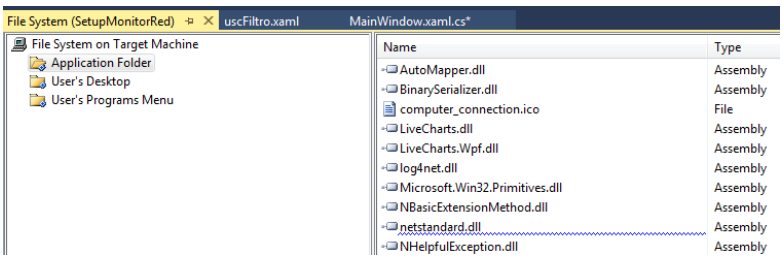


Ilustración 50. Captura 4 - Instalador aplicación

Desde “User’s Desktop” y User’s Programs se procede a cambiar el icono del acceso directo, el procedimiento es el mismo en los casos. El proceso para cambiar el icono del escritorio es el siguiente, hacemos botón derecho sobre “User’s Desktop”, “Crear nuevo acceso directo”. Cambiamos el nombre a MonitorRed y el icono en las propiedades del acceso directo.

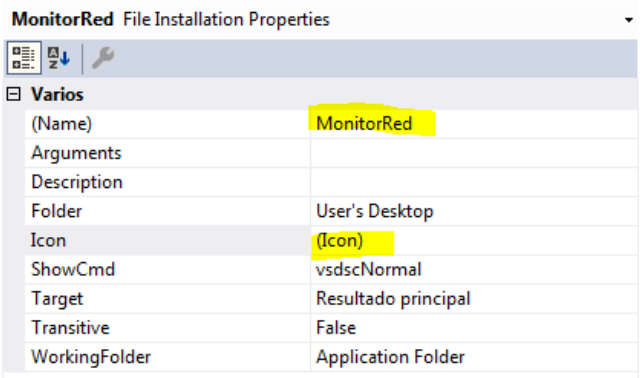


Ilustración 51. Captura 5 - Instalador aplicación

Por último, podemos cambiar opciones para personalizar la aplicación, como el autor, nombre compañía, versión, etc. Para ello, debemos marcar el proyecto creado, en nuestro caso, SetupMonitorRed.

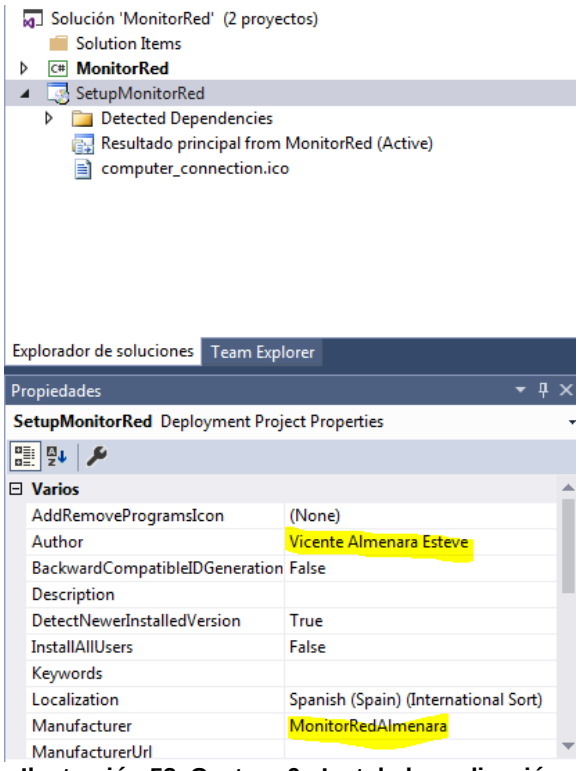


Ilustración 52. Captura 6 - Instalador aplicación

El ejecutable lo encontraremos en la carpeta donde esta nuestro proyecto.

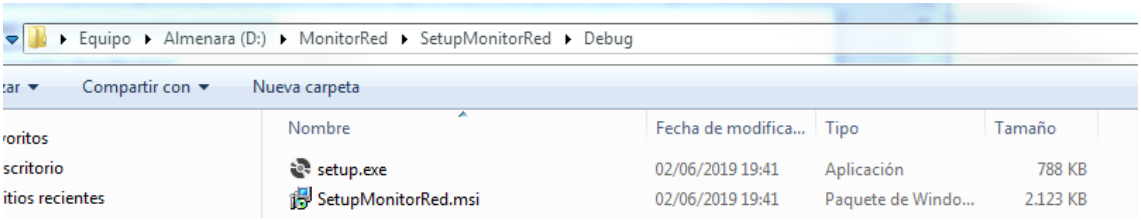


Ilustración 53. Captura 7 - Instalador aplicación

Para la instalación de la aplicación, ejecutamos el archivo setup.exe. Cuando finalice la instalación nos aparecerá en el escritorio y en el menú inicio el acceso directo, con el icono que hemos cambiado.