Herramienta de Análisis de Rendimiento de una Red

Ricardo Húmera Pulido
Grado de Ingeniería Informática

Consultora: María Isabel March Hermo

Enero 2020

C) Copyright

FICHA DEL TRABAJO FINAL

Título del trabajo:	Herramienta de análisis de rendimiento de una red
Nombre del autor:	Ricardo Húmera Pulido
Nombre de la consultora:	María Isabel March Hermo
Fecha de entrega:	01/2020
Área del Trabajo Final:	Redes de Computadores
Titulación:	*Grado de Ingeniería Informática*

Resumen del Trabajo (máximo 250 palabras):

¿Es óptimo el rendimiento de nuestra red?, ¿podemos aumentar el rendimiento?, ¿qué tipo de tráfico fluye por nuestra red?

Estas son preguntas en la que todo administrador de redes tendría que hacerse todos los días, aunque la sensación que tenga él y sus usuarios sea la inexistencia de problemas.

Para obtener respuestas a estas preguntas es necesario el uso de herramientas de análisis de red, que nos arrojan resultados de sus análisis para que los administradores de redes puedan sacar conclusiones y actuar respeto a estas.

Este Trabajo Fin de Grado (TFG) describe el desarrollo de creación de una herramienta de análisis de rendimiento de una red *ethernet*, con la que se pretende que un usuario pueda conocer el comportamiento de una red y ayudarle a sacar conclusiones de su tráfico.

Este análisis se basa en la “escucha” de la red donde está conectado y su posterior clasificación del tráfico de los protocolos utilizados en cada capa del modelo TCP/IP. Además, también presenta el análisis de conversaciones entre equipos pertenecientes a la red tanto en capa física como de red. También se ha realizado un módulo de estadísticas donde se presentan los datos recogidos en gráficas para un mejor análisis por parte del usuario.

El proyecto está compuesto por tres bloques. El primero consiste en un breve estudio del modelo TCP/IP y más concretamente como se estructura los datos. El segundo es la fase de análisis y diseño de la aplicación. El último bloque describe la implementación de la herramienta.

Abstract (in English, 250 words or less):

Is our network performance optimal?, can we increase performance?, what kind of traffic flows through our network?

These are questions that every network administrator should ask himself every day, although the feeling that he and his users have is there are no problems.

To obtain answers to these questions, it is necessary to use network analysis tools, which give us results of their analysis so that network administrators can draw conclusions and act on them.

This End-of-Degrade Project (TFG) describes the development of a performance analysis tool for an ethernet network, which is intended to enable a user to know the behavior of a network and help him or her draw conclusions from his traffic.

This analysis is based on the "listening" of the network where it is connected and subsequent classification of the traffic in the protocols used in each layer of the TCP/IP model. In addition, it also presents the analysis of conversations between computers belonging to the network at both the physical and network layers. A statistics module has also been created where the data collected is presented in graphs for better analysis by the user.

The project is composed of three blocks. The first consists of a brief study of the TCP/IP model and more specifically how the data is structured. The second is the analysis and design phase of the application. The last block describes the development of the tool.

Palabras clave (entre 4 y 8):

Sniffer, rendimiento, Python, ethernet, IP, TCP, UDP, redes

Índice

Lista de ilustraciones

1. Introducción

En estos tiempos actuales la utilización de dispositivos para el acceso a la información se ha convertido en uso cotidiano que está presente en casi todas las acciones del día a día. Esto nos ha llevado a abstraernos de su funcionamiento y únicamente exigimos resultados independientemente de la forma que estos son utilizados para su fin. Pero… ¿realmente esto es importante?

Desde un punto de vista de usuario final esto no es importante mientras que la satisfacción de funcionamiento sea de su agrado. Pero si trasladamos la situación a un negocio o compañía, este aspecto cobra mayor relevancia, ya que la eficiencia de funcionamiento afecta directamente a la productividad e incluso a la seguridad.

Como cualquier máquina (un coche, un reloj, una nevera, etc.) para saber si su funcionamiento es correcto o para aplicar una reparación, es necesario realizar un trabajo previo de obtención de datos para su posterior análisis y toma de decisiones. El uso de herramientas para este objetivo es fundamental y cuanto más eficientes sean estas, menor será el coste en la resolución de los problemas.

Además de conocer el rendimiento de una red también es necesario entender el estado de esta en función de las necesidades organizativas, de quién hace uso de la red, y cómo se está utilizando. En este sentido, las herramientas de análisis de red pueden ayudar aportar información a las políticas o auditorias de una compañía para modificar sus estrategias y resolver conflictos.

1.1 Contexto y justificación del Trabajo

En las redes de comunicaciones actuales se procesan enormes volúmenes de datos y cada vez tenemos nuevas formas de procesarlos. Tenemos una gran cantidad de protocolos y cada uno de ellos con un funcionamiento especifico. Los técnicos de redes y seguridad están obligados a utilizar herramientas que se adapten a la naturaleza de cada red que gestionan para dar soluciones de cualquier índole.

En este trabajo se pretende conseguir una herramienta de análisis de red destinada a técnicos de redes y seguridad que puedan realizar un análisis de rendimiento, y más específicamente que tipo de tráfico fluye en sus redes. La representación de los resultados aspira a que la toma de decisiones sea la más productiva.

Como finalización de los estudios del Grado de Ingeniería de Informática y como objetivo final, es poder realizar una síntesis de los conocimientos adquiridos en el transcurso del grado y ponerlos en práctica.

Con todo lo aprendido en estos años siempre me he encontrado muy cómodo en todo lo relacionado con redes y comunicaciones, por lo que no he tenido ninguna duda en realizar el TFG en la rama de Redes de Computadores.

El aspecto de más peso a la hora de decantarme por el desarrollo de una aplicación de análisis de redes ha sido el querer profundizar en la manera que tienen las redes en funcionar a bajo nivel. Adquirir conocimientos en este aspecto te aseguran entender otras áreas de las comunicaciones e incluso resolver problemas con mayor dificultad.

1.2 Objetivos del Trabajo

El objetivo es realizar una aplicación gráfica de escritorio la cual nos permitirá capturar el tráfico que fluye por una red, clasificarlo, contabilizarlo y representarlo en gráficas. Estos tres últimos aspectos deben ayudar al usuario de la herramienta a conseguir un conocimiento del estado la red y presentarle los datos para que pueda sacar sus propias conclusiones.

La aplicación se basa en tres bloques:

- Captura de tráfico
- Clasificación y contabilización del tráfico
- Presentación de resultados.

La captura de datos tiene que ser optima y con mínima perdida de información, por lo que se podrán en práctica técnicas de programación multiproceso para sacar el máximo rendimiento al hardware además de permitir la obtener dataos procedentes de cualquier *inteface* de red que esté disponible en el sistema.

La clasificación se debe ceñir al modelo TCP/IP, donde según el tráfico capturado se marcará con su tipo, capa y protocolo correspondiente. Se contabilizarán el número de paquetes del protocolo marcado además de su tamaño. La visualización de esta información debe representarse de una manera cómoda y atractiva para el usuario.

Una vez clasificado este tráfico, se trasladarán los datos a gráficas. Estas gráficas serán de modelo de barras o tarta, representado el tráfico por el tipo de capa del modelo TCP/IP y protocolo. Asimismo, se podrá seleccionar si estas gráficas sean por número de paquetes analizados o por tamaño en Bytes.

Como último objetivo se pretende que la herramienta sea multiplataforma y obtener varias distribuciones para ofrecer al usuario la posibilidad de ejecutarla en diversos sistemas operativos.

1.3 Enfoque y método seguido

Para la realización del proyecto se ha seguido un método de documentación, análisis y elaboración que se divide en cuatro fases.

La primera se basa en la recopilación de información, donde ha habido un estudio previo de protocolos y las capas de red del modelo TCP/IP. Además, también se han buscado aplicaciones en el mercado de análisis de redes para enfocar la construcción de la herramienta.

La segunda fase consiste en el análisis y diseño de la aplicación. Aquí se han definido los requisitos, casos de uso, diagrama de clases, etc. También se ha realizado un pequeño estudio del entorno de desarrollo y la decisión de qué queremos analizar en la red.

Una vez decido el entorno y lenguaje de programación, se pasa a la fase de implementación, donde primero se elaboró un módulo de captura y seguidamente se implementó la parte gráfica GUI (*Graphical User Interface*).

La siguiente fase define qué información se representa y cómo se representa. Es una fase que se solapa con a la anterior ya que hay una carga importante de desarrollo de programación para la elaboración de las gráficas de estadísticas.

Por último, queda la fase de pruebas y corrección de errores donde se han comparado los resultados con otras aplicaciones de análisis de redes consolidadas.

La parte de la elaboración de la memoria ha sido trasversal a todas las fases descritas anteriormente, donde se ha ido ampliando y modificando cada vez que se superaba alguna de ellas.

1.4 Planificación del Trabajo

La planificación del trabajo se divide en 7 tareas y se fijan en calendario correspondiente al segundo semestre de 2019. En cada tarea se define la duración y los objetivos a cumplir.

Tarea 1

Duración: 1 semana (30 de septiembre a 6 de octubre)

Descripción: Recogida y clasificación de información. Analizar herramientas similares de analizadores de rendimiento de redes. Estudio de protocolos que analizaremos en la herramienta:

- Protocolos capa Física: ethernet
- Protocolos capa Red: IP, ARP

- Protocolos capa Transporte: TCP, UDP, ICMP
- Protocolos capa Aplicación: HTTP, FTP, NTP, etc.

Objetivos: El estudio de estos protocolos nos dará una visión general de cómo se debe recoger la información de la red y seleccionaremos los protocolos de la capa de aplicación más importantes para su análisis. Además, analizaremos un par herramientas similares.

Tarea 2

Duración: 3 semanas (7 de octubre al 27 de octubre)

Descripción: Análisis y diseño:

- Análisis de requisitos
- Diseño de casos de uso
- Selección del entorno de trabajo (lenguaje de programación)
- Estructurar documento de memoria

Objetivos: En esta fase realizaremos el análisis de requisitos y los casos de uso. En la parte de selección de entorno de trabajo realizaremos un estudio de los lenguajes más adecuados para el aplicativo y se seleccionará uno. Por último, empezaremos a realizar la memoria y su estructura.

Tarea 3

Duración: 2 semana (28 de octubre al 10 de noviembre)

Descripción: Implementación captura de datos

- Estudio de librería para la captura de datos
- Implementación no gráfica de la captura de datos
- Organización de los datos
- Actualización de memoria

Objetivos: En esta fase se probarán la librería para la captura de datos y se realizará la implementación no gráfica. Con ello seremos capaces de realizar un análisis de los datos capturados y proponer una estructura para el entorno gráfico.

Tarea 4

Duración: 2 semanas (11 de noviembre al 24 de noviembre)

Descripción: Implementación parte gráfica.

- Realización de la parte gráfica de la aplicación
- Pruebas de funcionamiento y corrección de errores

Objetivo: Una vez que sabemos cómo están capturados los datos la representación gráfica se imprentará para una lectura correcta para el usuario.

Tarea 5

Duración: 2 semanas (25 de noviembre al 8 de diciembre)

Descripción: Implementación estadísticas

- Estudio de librerías para representación gráfica de estadísticas
- Realización de la parte de estadísticas en modo gráfico
- Pruebas de funcionamiento y corrección de errores

Objetivo: Seleccionar una librería gráfica para la representación gráfica de los datos capturados. Implementación de parte de estadísticas gráficas.

Tarea 6

Duración: 1 semana (9 de diciembre al 15 de diciembre)

Descripción: Pruebas de funcionamiento y corrección de errores.

Objetivo: Realizar una batería de pruebas para descubrí posibles errores y subsanarlos antes de su versión definitiva. Creación de programas de instalación y manuales de usuario.

Tarea 7

Duración: 3 semanas (16 de diciembre al 5 de enero)

Descripción: Correcciones finales de la memoria del TFG para su presentación

Objetivo: Organizar toda la información que se ha ido recopilando durante la elaboración del TFG para su perfecta presentación y entrega.
Durante las dos primeras tareas no habido desviaciones en la planificación, ya que principalmente era la recopilación y ordenación de información. La parte de la selección del lenguaje de programación e IDE (*Integrated Development Environment*) ha sido trasversal a las tares 2 y 3.

La parte donde he tenido más problemas ha sido en la fase implantación y en concreto, en la clasificación del tráfico capturado. En los objetivos iniciales del proyecto se definía que únicamente se analizaría los protocolos más conocidos en la capa de aplicación (HTTP, HTTPS, DNS, NTP o FTP), pero a medida que

ido avanzado veía que está clasificación se quedaba corta, y que la herramienta no conseguiría los objetivos principales de ofrecer al usuario un análisis correcto del estado de la red.

La decisión de clasificar la máxima cantidad de protocolos llevó al proyecto a ampliar fechas en la fase de implantación, ya que conllevaba a profundizar más en el lenguaje de programación y del GUI. El impacto de este cambio ha sido de analizar unos 10 protocolos a casi 350 de ahora, incluso en la fase de pruebas se han añadido nuevos protocolos.

Otra parte donde he tenido problemas ha sido en la tarea 6 en la creación de versiones para diferentes sistemas operativos. Aquí he tenido que realizar cambios en el código para que este esté preparado para las diversas plataformas con la consiguiente ampliación de días en dicha tarea.

1.5 Breve sumario de productos obtenidos

Los productos obtenidos en este trabajo son los siguientes:

- Memoria del Trabajo Fin de Grado
- Herramienta de análisis para su ejecución (AnaLiz)
- Código en Python
- Manual de usuario (incluido en este documento)

1.6 Breve descripción de los otros capítulos de la memoria

Conceptos: Fase de conceptualización. Breve estudio sobre el modelo TCP/IP y como están construidos. Pequeño estudio de herramientas similares del mercado.

Análisis y diseño: Definición de requisitos, casos de uso, diagrama de clases y GUI.

Implementación: Descripción del desarrollo de la aplicación, así como sus retos y dificultades.

Manual: Descripción de requisitos para su ejecución y manual de uso.

Conclusión: Conclusiones y valoraciones sobre el conjunto del TFG.

2. Conceptos

En este bloque de conceptualización analizaremos el modelo de referencia OSI y el modelo TCP/IP. A continuación, se detallarán los protocolos base en las comunicaciones TCP/IP: Ethernet, ARP, ICMP, IP.

Además, como protocolos fundamentales en la gestión de las aplicaciones se detallarán los protocolos TCP y UDP, y una selección de protocolos de la capa de aplicación.

La aplicación que estamos desarrollando tiene un apartado que clasifica el tipo de tráfico *Broadcast*, *Multicast* y *Unicast*, por lo que también se hace mención de estos conceptos.

Por último, se hará un pequeño estudio de herramientas similares en el mercado para analizar como representan los datos.

2.1 Modelo de referencia OSI

A principios de los años ochenta las principales empresas de fabricación de equipos de informática se reúnen para unificar criterios de cómo poder integrar sus productos para conseguir que sean compatibles y comunicarse entre ellos. Como resultado surge el modelo de referencia llamado Modelo de Interconexión de Sistemas Abiertos OSI (*Open System Interconnection*) que define los parámetros comunes de hardware y software haciendo posible la integración multifabricante.

El modelo OSI divide a una red en 7 capas:

Ilustración 1. Modelo OSI

Cada capa presta servicio a la capa inmediatamente superior, donde la capa de aplicación es la única que no lo hace ya que al ser la última su servicio está directamente relacionado con el usuario. Por tanto, cada una de estas siete capas se comunican directamente desde un origen con su similar en un destino.

2.2 Modelo TCP/IP

En la actualidad con a la expansión de las redes basadas en TCP/IP el modelo de referencia pasa de 7 capas a 4, donde se agrupan varias capas de OSI en una sola. Este modelo pasa a ser denominado Modelo TCP/IP.

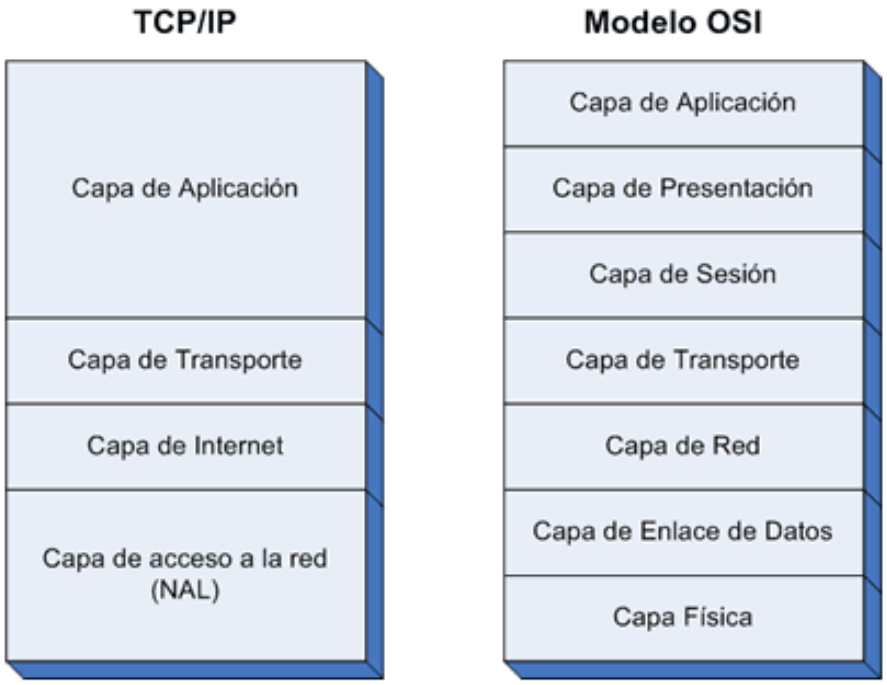


Ilustración 2. Comparación entre modelo OSI y TCP/IP

En algunas referencias la capa de Acceso a Red del modelo TCP/IP se continúa segregando en las capas de Enlace y capa Física, pero continúan con la misma función.

Capa de Acceso a la Red

Esta capa corresponde a las capas física y de enlace del modelo OSI. La capa física comprende a los medios físicos, conectores y tipos de señalización. Es la encargada de transportar y propagar los datos por lo que no los manipula y se rigen por unos estándares definidos.

También en la Capa de Acceso a la Red está incluida la capa de enlace cuya finalidad es proporcionar comunicación (a través de direccionamiento) entre los

distintos dispositivos de red. Esta comunicación se realiza por medio de protocolos donde el protocolo Ethernet es el más utilizado.

Capa de Internet

Controla la comunicación entre dispositivos origen y destino de red que se utilizan para enviar los datos. La determinación de la mejor ruta ocurre en esta capa. Los 3 protocolos importantes en esta capa son IP, ARP e ICMP.

Capa de Transporte

Es la encarda de proveer comunicación de extremo a extremo desde una aplicación a otra. Puede asegurar que los datos se envíen y lleguen sin errores y en una secuencia correcta.

Los protocolos usados son TCP y UDP.

Capa de Aplicación

Esta capa es la que define los servicios que utiliza el usuario, es decir, maneja los protocolos de alto nivel. En esta capa se combinan todos los aspectos relacionados con las aplicaciones y asegura que estos datos estén correctamente empaquetados antes de que enviados.

Hay infinidad de protocolos, pero unos de los más conocidos son HTTP, HTTPS, FTP, DNS, SMTP…

Encapsulamiento de capas

Al igual que en modelo OSI, en cada fase de se añade a los datos a enviar una información de control con propiedades de la propia capa. Esta información añadida se la denomina cabecera y la acción de realizarlo en cada capa se la denomina encapsulamieno. El proceso contrario se realiza a la inversa, se elimina la cabecera para pasar a su vecina.

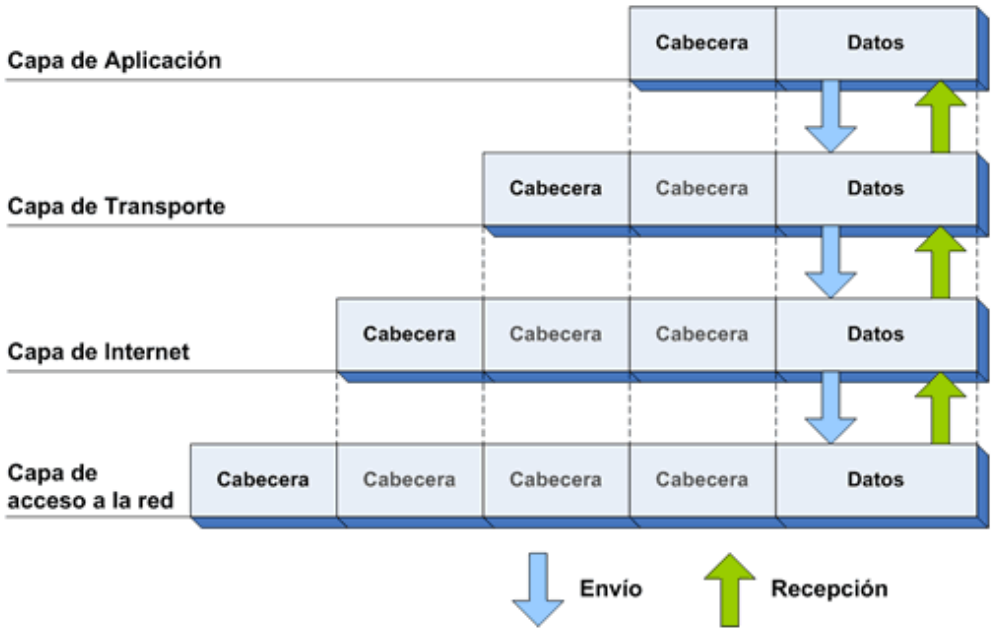


Ilustración 3. Formación de cabeceras entre capas

2.3 Ethernet

Ethernet es la tecnología de acceso al medio más popular, es escalable, económica y fácilmente integrable a nuevas aplicaciones. La forma en que las redes transmiten los datos es a través ethernet y describe su formato en tramas. Ethernet opera sobre la capa de Acceso a la Red del modelo TCP/IP.

La estructura de la trama Ethernet añade cabeceras alrededor de la PDU de capa 3 para encapsular el mensaje enviado. Existen dos tipos de entramado el IEEE 802.2 (el original) y el IEEE 802.3 revisado.

La diferencia entre ambos es mínima. La más significativa es la incorporación de un SFD (Delimitador de inicio de trama, *Start Frame Delimiter*) y un pequeño cambio en el campo Tipo para incluir la longitud.

Preámbulo	Delimitador de inicio de trama	MAC de destino	MAC de origen	Etiqueta 802.1Q (opcional)	Longitud /Tipo	Encabezado y datos	Secuencia de verificación de trama (FCS)
7 Bytes	1 Bytes	6 Bytes	6 Bytes	4 Bytes	2 Bytes	46 a 15000 Bytes	4 Bytes

Ilustración 4. Estructura Ethernet

El estándar original definía el tamaño mínimo de la trama en 64 bytes y el máximo en 1518 bytes. Esto incluía todos los bytes desde el campo de dirección MAC de destino hasta la FCS (Secuencia de Verificación de trama, *Frame Check*

Sequence). Los campos Preámbulo y SFD no se incluyen cuando se describe el tamaño de una trama.

Más adelante se publica el estándar IEEE 802.3ac que amplia el tamaño máximo de la trama a 1522 bytes. Este aumento es hizo para dar capacidad a la tecnología que usaban las redes VLAN (Red de área local virtual, *Virtual Local-Area Network*).

Si el tamaño de una trama recibida es menor que el valor mínimo o mayor que el máximo se considera una trama corrompida y será descartada.

2.4 Protocolo ICMP

El protocolo ICMP (*Internet Control Message Protocol*) es un protocolo que permite suministrar información (mensajes de error y control) sobre problemas relacionados con el procesamiento de paquetes IP bajo ciertas condiciones, pero no para conseguir que las comunicaciones IP sean fiables, ya que de esto se encargan las capas superiores.

ICMP siendo un protocolo de capa de Acceso a Red forma parte del conjunto de protocolos IP, por lo que se ubicará en la capa de Internet en el modelo TCP/IP. La cabecera ICMP es la siguiente:

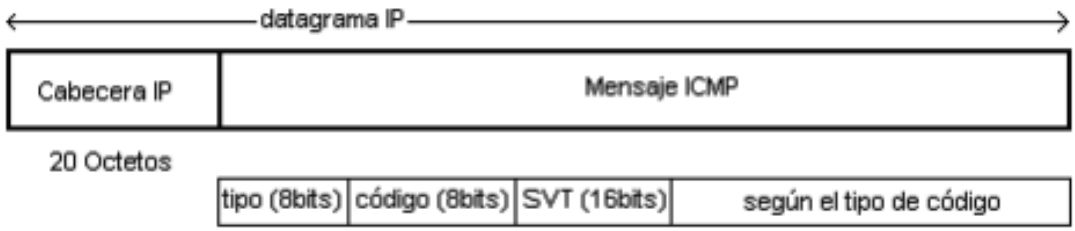


Ilustración 5. Estructura protocolo ICMP

La combinación de tipos y códigos de mensajes ICMP son amplios y pueden referirse a la confirmación de host, destino o servicio inalcanzable, tiempo excedido, redirección de ruta o origen apagado.

2.5 Protocolo ARP

El protocolo ARP (*Address Resolution Protocol*) juega un papel importante en las comunicaciones TCP/IP. El principal objetivo es conocer las direcciones físicas (MAC) de los elementos que participan en una red, y relacionarla con la dirección IP.

Para este objetivo una máquina envía un paquete (*ARP request*) a una dirección *broadcast* con la dirección IP por la que pregunta. Esta se queda esperando a recibir de otra máquina un paquete (*ARP reply*) con la dirección MAC.

Cada máquina crea su propia tabla ARP con las direcciones traducidas para reducir el retardo y la carga. ARP permite a la dirección IP ser independiente de la dirección MAC.

El protocolo ARP trabaja entre las capas Física y de Acceso a Red, por lo que es posible que en algunas publicaciones la encuadren en una de estas dos capas.

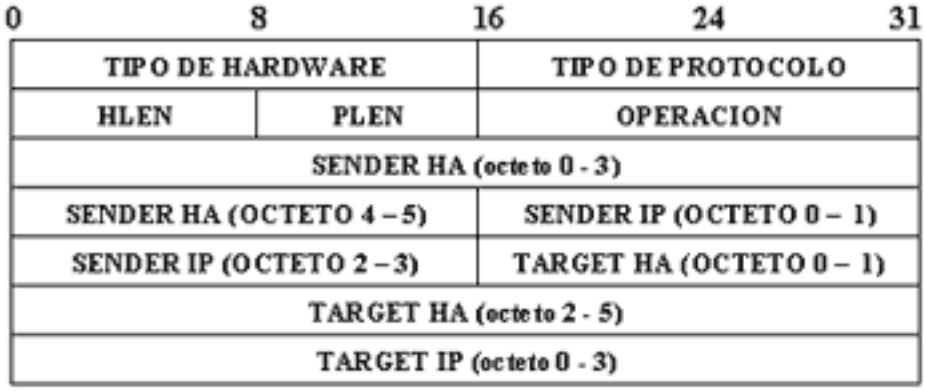


Ilustración 6. Estructura protocolo ARP

2.6 Protocolo IP

El protocolo IP (*Internet Protocol*) pertenece a la capa de red y gracias a su direccionamiento habilita la comunicación de datos entre dispositivos pertenecientes a la misma red o en redes diferentes y garantiza que las redes funcionan de forma eficaz y eficiente.

Cada dispositivo de una red debe estar definido en exclusiva por dirección de la capa de red. En esta capa, los paquetes en la comunicación también están identificados con direcciones de origen y destino. Con IPv4 cada paquete utiliza una dirección origen de 32 bits y una dirección destino también de 32 bits. IPv6 utiliza direcciones de tamaño de 128 bits.

El datagrama IPv4 es el paquete de transferencia base:

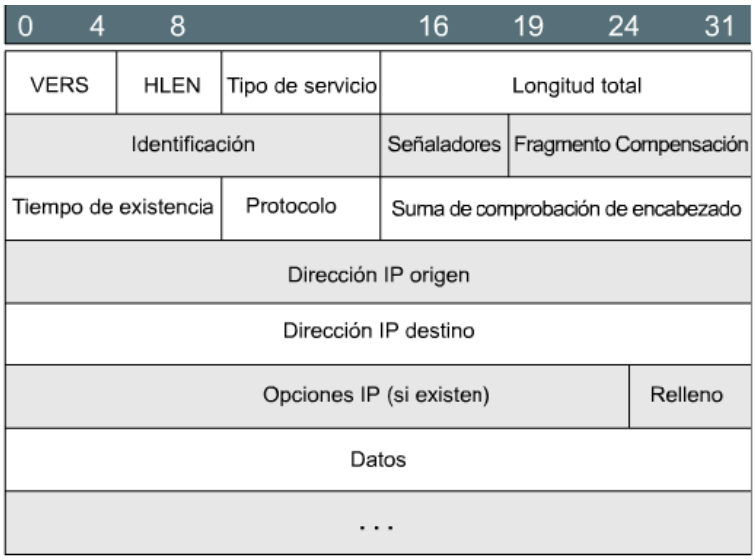


Ilustración 7. Estructura protocolo IP

2.7 TCP y UDP

TCP (*Transfer Control Protocol*) y UDP (*User Datagram Protocol*) son los dos protocolos más comunes en la capa de Internet. Estos protocolos gestionan la comunicación de las aplicaciones. Para referenciar estas aplicaciones se utilizan números de puerto que permiten a una máquina establecer simultáneamente múltiples conexiones con otras máquinas, ya que la conexión se recibe en la misma dirección, pero van dirigidos a puertos diferentes.

En una comunicación se puede utilizar cualquier número, pero existe una asignación global que relaciona estos números a las aplicaciones, para evitar las incompatibilidades en las comunicaciones. IANA (www.iana.org) es la organización que se encarga de este trabajo.

TCP

TCP posibilita la comunicación de datos entre equipos y posibilita la administración de datos que viene de su nivel más bajo, es decir, el protocolo IP. Cuando se proporcionan los datos al protocolo IP, lo agrupa en datagramas IP y fija el campo del protocolo en 6 para anticiparse y saber que se trata de protocolo TCP. Es un protocolo orientado a conexión por lo que permite que dos máquinas estén comunicadas y controlen el estado de la transmisión.

La principal característica de TCP es poner nuevamente en orden los datagramas procedentes del protocolo IP. También permite que los datos se formen en segmentos de longitud variada para que el protocolo IP permita entregarlos y que estos se puedan multiplexar para que la información procedente de diversas fuentes pueda circular simultáneamente. Por último, TCP es la encargada comenzar y finalizar una comunicación y además está

preparada para continuar con ella, aunque se produzcan cortes de comunicación en la red.

Un segmento TCP está formado de la siguiente manera:

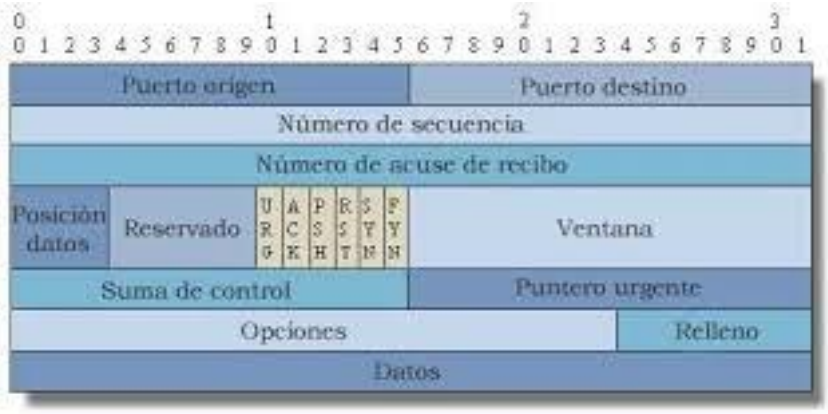


Ilustración 8. Estructura protocolo TCP

UDP

UDP es un protocolo sencillo y no orientado a la conexión por lo que proporciona una entrega de datos con poca sobrecarga. Esto quiere decir que cuando la máquina A quiere enviar paquetes a una máquina B este flujo es en un único sentido.

La máquina A envía la información sin que previamente se establezca una conexión y la máquina receptora recibirá los datos sin la necesidad de que está comunique la recepción a la máquina A. Por lo que con UDP los datos son enviados sin saber si van a ser recibidos correctamente, el orden o si están completos.

Bits	0 - 15	16 - 31
0	Puerto origen	Puerto destino
32	Longitud del Mensaje	Suma de verificación
64	Datos	

Ilustración 9. Estructura protocolo UDP

En la siguiente tabla se representa las diferencias entre TCP y UDP:

TCP	UDP
Es un servicio orientado a la conexión entre dos máquinas	Es un servicio que no está orientado a la conexión
El protocolo garantiza la entrega y su secuencia de datos	El protocolo no garantiza ni confirma ni la entrega ni la secuencia de datos
Solo admite la comunicación punto a punto	Permite la comunicación punto a punto o punto a varios destinos
Lento en transporte de datos (relativamente)	Rápido en el transporte de datos

2.8 Protocolos capa ampliación

En la capa de ampliación trabajan números protocolos y es difícil definir una selección de los más importantes. No obstante, en la siguiente lista se hacen referencia alguno de ellos:

- SSH: Este protocolo proporciona el acceso remoto a un host por medio de un canal seguro en el que toda la información está cifrada.
- DNS: Es utilizado en internet para convertir los nombres de los nodos de red en direcciones.
- SMTP: Es utilizado para el intercambio de mensajes de correo electrónico entre distintos dispositivos. Se base en el modelo cliente-servidor, donde un cliente envía un mensaje a uno o varios receptores.
- SNTP: Es utilizado casi con exclusividad en redes TCP/P. En SNMP brinda una forma de monitorizar y controlar los dispositivos de red y administrar configuraciones.
- DHCP: Este protocolo proporciona mecanismos para asignar direcciones IP de forma dinámica, de modo que las direcciones se pueden reutilizar automáticamente cuando los hosts ya no las necesitan.
- HTTPS: Este protocolo de transferencia de hipertexto destinado a la transferencia segura de datos de hipertexto que ofrece un canal seguro y toda la información está cifrada.

2.9 Tráfico Broadcast, Multicast y Unicast

El tráfico *broadcast* utiliza una dirección especial que permite a todos los dispositivos de una red aceptar y procesar la trama des dispositivo origen. En ethernet la dirección MAC de broadcast está compuesta por los 48 unos en hexadecimal FF:FF:FF:FF:FF:FF:FF:FF.

Una dirección *multicast* permite a un dispositivo de origen enviar un paquete a un grupo de dispositivos. Estos dispositivos destinos están asignados a una dirección IP del grupo de *multicast*. El rango de las direcciones IPv4 *multicast* van desde la 224.0.0.0 hasta la 239.255.255.255.

Cuando un dispositivo desea participar en un grupo *multicast* usa una aplicación o un servicio para suscribirse al mismo. La dirección IPv4 del *multicast* requiere la correspondiente dirección MAC *multicast* para entregar las tramas en una red.

La dirección MAC *multicast* es un valor especial que comienza con 01:00:5E en hexadecimal. El valor finaliza convirtiendo los 23 bits inferiores de la dirección de grupo *multicast* IP en los restantes seis caracteres hexadecimales de la dirección Ethernet. El bit restante de la dirección MAC es siempre 0.

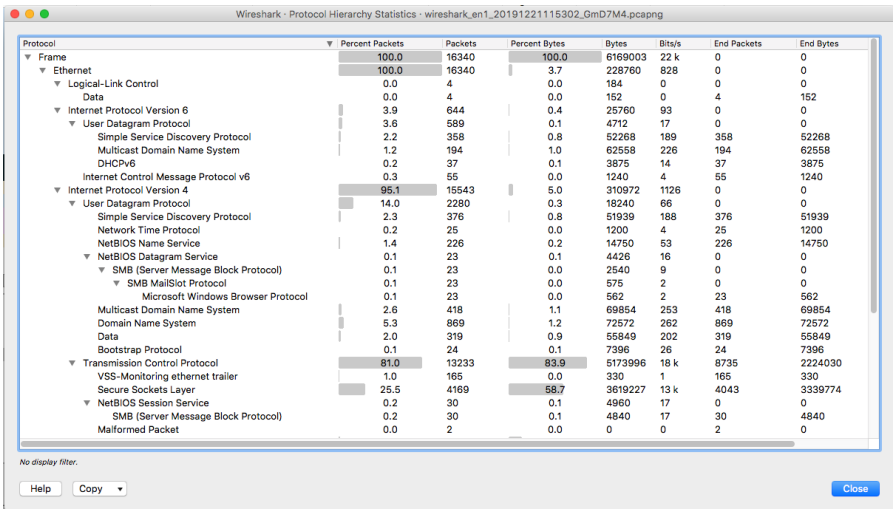
Una dirección MAC de *unicast* es la dirección única que se utiliza cuando una trama se envía desde un único dispositivo a otro dispositivo.

2.10 Aplicaciones similares como referencia

Como primer paso es crear una primera idea de cómo se quiere presentar la información, para ello se ha realizado un pequeño ejercicio de estudio con algunas aplicaciones dedicadas al análisis del tráfico de red. Aplicaciones como WireShark, Orion de SolarWinds o Capsa de Colasoft.

La forma de capturar en el caso de WireShark y Capsa es muy parecida: es un *sniffer* que captura tráfico por una tarjeta de red. En el caso de Orion, la captura de datos se realiza por los protocolos SNMP y NetFlow. Esto significa que habría que tener configurado en cada *host* de la red estos dos servicios, por lo WireShark y Capsa se acercan más al propósito de nuestra herramienta.

A continuación, se muestran capturas de pantalla de estas tres aplicaciones:



Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes
Frame	100.0	16340	100.0	6169003	22 k	0	0
Ethernet	100.0	16340	3.7	228760	828	0	0
Logical-Link Control	0.0	4	0.0	184	0	0	0
Data	0.0	4	0.0	152	0	4	152
Internet Protocol Version 6	3.9	644	0.4	25760	93	0	0
User Datagram Protocol	3.6	589	0.1	4712	17	0	0
Simple Service Discovery Protocol	2.2	358	0.8	52268	189	358	52268
Multicast Domain Name System	1.2	194	1.0	62558	226	194	62558
DHCPv6	0.2	37	0.1	3875	14	37	3875
Internet Control Message Protocol v6	0.3	55	0.0	1240	4	55	1240
Internet Protocol Version 4	95.1	15543	5.0	310972	1126	0	0
User Datagram Protocol	14.0	2280	0.3	18240	66	0	0
Simple Service Discovery Protocol	2.3	376	0.8	51939	188	376	51939
Network Time Protocol	0.2	25	0.0	1200	4	25	1200
NetBIOS Name Service	1.4	226	0.2	14750	53	226	14750
NetBIOS Datagram Service	0.1	23	0.1	4426	16	0	0
SMB (Server Message Block Protocol)	0.1	23	0.0	2540	9	0	0
SMB MailSlot Protocol	0.1	23	0.0	575	2	0	0
Microsoft Windows Browser Protocol	0.1	23	0.0	562	2	23	562
Multicast Domain Name System	2.6	418	1.1	69854	253	418	69854
Domain Name System	5.3	869	1.2	72572	262	869	72572
Data	2.0	319	0.9	55849	202	319	55849
Bootstrap Protocol	0.1	24	0.1	7396	26	24	7396
Transmission Control Protocol	81.0	13233	83.9	5173996	18 k	8735	2224030
VSS-Monitoring ethernet trailer	1.0	165	0.0	330	1	165	330
Secure Sockets Layer	25.5	4169	58.7	3619227	13 k	4043	3339774
NetBIOS Session Service	0.2	30	0.1	4960	17	0	0
SMB (Server Message Block Protocol)	0.2	30	0.1	4840	17	30	4840
Malformed Packet	0.0	2	0.0	0	0	2	0

Ilustración 10. Jerarquía de Protocolos Wireshark

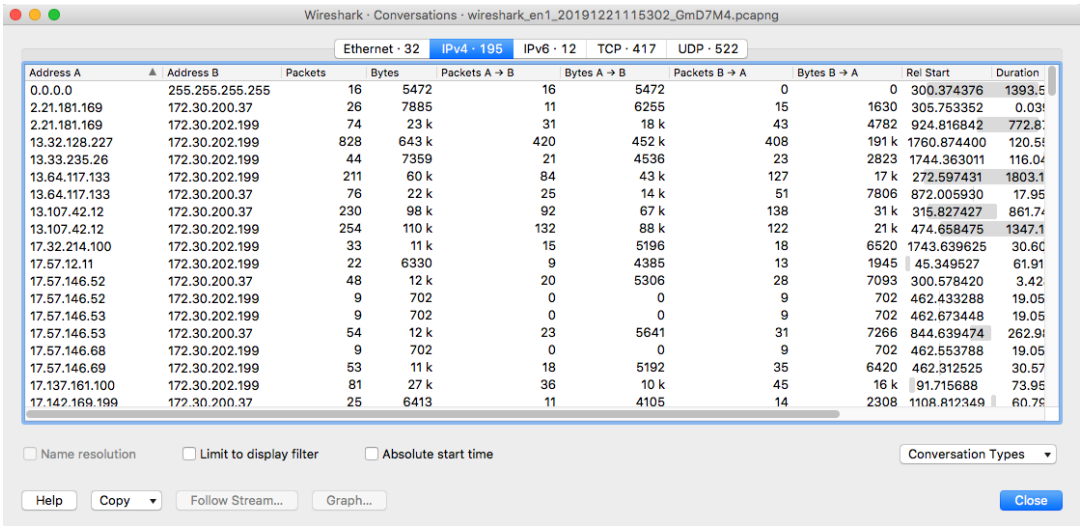


Ilustración 11. Conversaciones IPv4 Wireshark

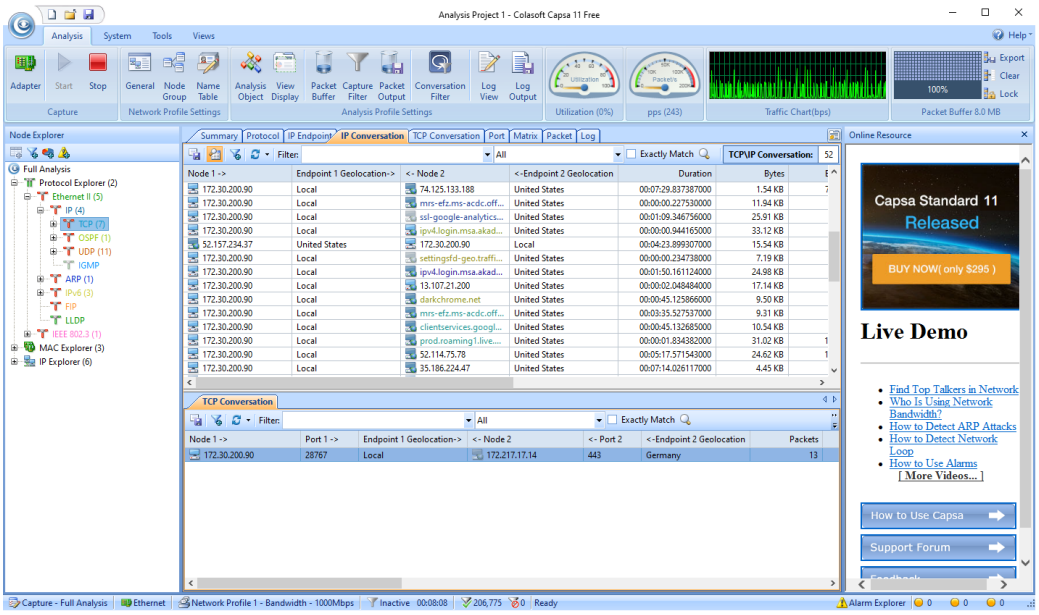


Ilustración 12. Jerarquía de protocolos y conversaciones IPv4 Capsa de Colasoft

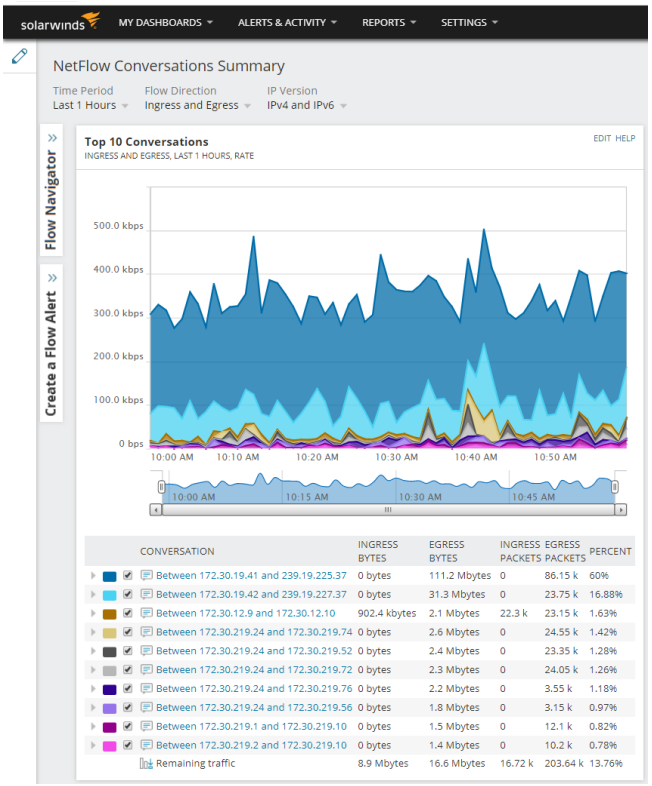


Ilustración 13. Conversaciones IPv4 Orion SolarWinds

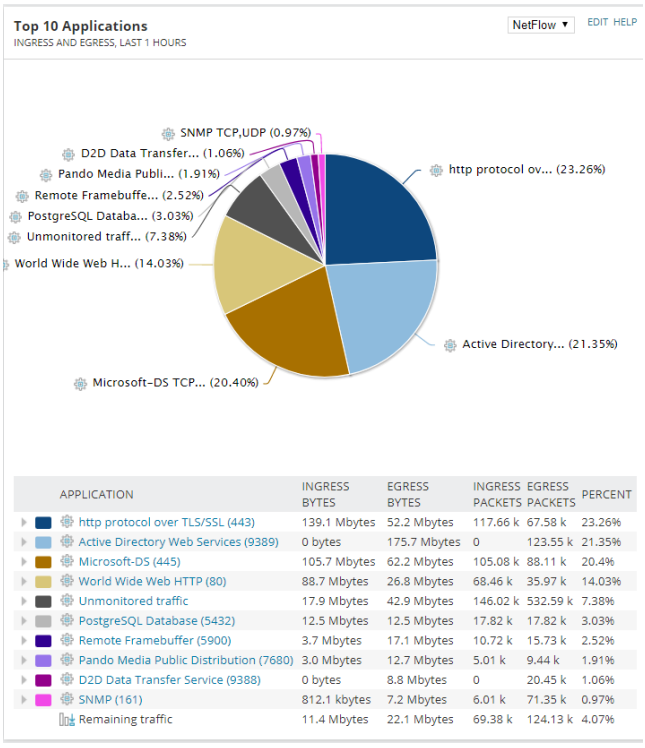


Ilustración 14. Top aplicaciones Orion SolarWinds

3. Análisis y diseño

3.1 Especificaciones de la herramienta

Esta herramienta de Análisis de Rendimiento de Red para este TFG tendrá la misión de presentar datos de una red a donde está conectada, para que el usuario que la utilice pueda ayudarle a conocer el comportamiento que tiene la red respecto a los datos que circulan por ella.

La aplicación tiene cuatro bloques:

- Captura de datos
- Clasificación y contabilización de datos
- Representación de datos
- Representación de estadísticas

En la captura de datos la aplicación dará la opción al usuario de que *interface* de red será la encarga de recoger los datos.

La clasificación de datos consiste en identificar el tráfico e*thertnet* de la red. Todo tráfico que no sea de este tipo no será catalogado. A partir de aquí, la herramienta empezará a clasificar y contabilizar dependiendo del **tipo de tráfico** (*unicast, multicast o broadcast*), de los **protocolos de capa de Red** (ARP, IPv4, IPv6 etc.), de los **protocolos de capa de Internet** (TCP, UDP, ICMP, etc.) y/o de los protocolos de la **capa de Aplicación** (HTTP, HTTPS, SSH, FTP, DNS, NTP, etc.).

Además, se mostrarán las conversaciones entre hosts, tanto a nivel *ethernet* como a nivel IPv4. En estas conversaciones se contabilizará el tráfico total como el sentido de este (de *Host* A -> *Host* B y viceversa).

La representación de estadísticas mostrará toda la información general clasificada con contadores independientes tanto por número de paquetes como por tamaño de estos en Bytes. Esta clasificación será la siguiente:

- Total del tráfico analizado
- Conversaciones *ethernet*
- Conversaciones IPv4
- Tráfico *Broadcast*
- Tráfico *Multicast*
- Tráfico *Unicast*

Por último, todos los datos recogidos se representarán visualmente mediante una serie de gráficas en ventanas independientes. Habrá tres bloques:

- **Gráficas generales:** Se representarán datos a nivel general en formato tarta del tipo de tráfico (*broadcast, multicast, unicast*), tráfico *ethernet*, tráfico IPv4, tráfico IPv6, tráfico TCP y tráfico UDP.

- **Gráficas de conversaciones:** Se representarán las conversaciones ethernet e IPv4 en gráficos tipo barra.

- **Gráficas de protocolos**: Se representarán cualquier protocolo analizado en formato tarta.

3.2 Requisitos

La aplicación debe cumplir una serie de características para su posterior análisis:

Requisito	Detalle
Elección de tarjeta de red	El usuario tendrá la obligación de seleccionar una interfaz de red para realizar la captura de tráfico
Captura de tráfico	Se debe captura de tráfico y esta debe procurar no tener pérdidas de paquetes
Clasificación del tráfico	Se realizará un análisis de la trama capturada para su posterior contabilización. Esta clasificación estará vinculada al tipo de protocolo. Además, se clasificará por tipo de tráfico (*Unicast, Multicast y Broadcas*t).
Contabilización del tráfico	Se contabilizará el tráfico clasificado por número de paquetes y tamaño en Bytes atendiendo a la clasificación.
Visualización de tráfico capturado	El sistema deberá representar los datos mediante dos sistemas: Jerarquía de protocolos, donde la información se representará mediante un árbol aplicando la clasificación de protocolos del modelo TCP/IP Conversaciones, donde se representará las conversaciones que hay en la red tanto a nivel *Ethernet* como de IPV4.

Estadísticas de tráfico	Con toda la información del tráfico capturado, el usuario tendrá la opción de poder analizar el tráfico mediante gráficas. Las gráficas se basarán en los protocolos más frecuentes en una red para determinar la salud de esta.
Multiplataforma	La herramienta deberá ser capaz de ejecutarse en las plataformas hardware más extendidas del mercado. Como mínimo se deberá ejecutarse en sistemas Microsoft Windows y macOS.

3.3 Diagrama de clases

A continuación, se presenta el diagrama de clases de entidad junto con sus atributos y métodos principales.

Sniffer: Clase para la captura de datos. La relación con la clase Main es asociativa, ya que al ser un hilo de ejecución es independiente una de otra.

- Stop(): Método privado para la parada del hilo de captura
- Run(): Método privado para la ejecución del hilo de captura.

Main: Clase para la clasificación y visualización del tráfico capturado.

- list_Hierarchy: lista de acceso pública donde se almacena los registros de la jerarquía de protocolos.
- list_conversations_Ether: Lista de acceso público donde se almacena las conversaciones Ethernet
- list_Conversations_IPv4: Lista de acceso público donde se almacena las conversaciones Ethernet
- list_DataGlobal: Lista de acceso público donde se almacena los datos de tráfico globales
- capture_interface: Inteface seleccionado por el usuario de acceso público.
- Started_callback(): Método privado para el retorno no hilo de captura en modo ejecución.
- finished_callback(): Método privado para el retorno no hilo de captura en modo parada.
- data_callback(): Método privado para el retorno datos (paquetes) del hilo de captura.
- clasificar_tráfico(): Método privado para clasificar un paquete.
- Insertar_jerarquía(): Método privado para inserción de datos en el widget del árbol de protocolos y actualización de list_Hierarchy.

- Insertar_Conversación_Ethernet(): Método privado para insertar conversación Ethernet en widget conversaciones y actualizar list_conversations_Ether.
- Insertar_Conversación_IPv4(): Método privado para insertar conversación IPv4 en widget conversaciones y actualizar list_conversations_IPv4.
- Reset(): Método privado de restaurar a valores iniciales todos los atributos de la clase Main
- Exportar_Captura(): Método privado para exportar los datos de captura a un archivo en disco.

Gráfica Generales: Clase para la visualizar gráfica de datos generales. Tiene dependencia de la clase Main ya que instancia una ventana independiente del formulario.

- List: Lista formateada para datos para la gráfica
- Generar_Lista_Datos():Método privado para crear la lista de datos.
- Imprimir_Gráfica(): Método privado para imprimir por pantalla las gráfica.
-

Gráfica Conversaciones: Clase para la visualizar gráfica de barras de conversaciones Ethernet o IPv4. Tiene dependencia de la clase Main ya que instancia una ventana independiente del formulario.

- List: Lista formateada para datos para la gráfica
- Generar_Lista_Datos():Método privado para crear la lista de datos.
- Imprimir_Gráfica(): Método privado para imprimir por pantalla las gráfica.

Gráfica Protocolos: Clase para la visualizar gráfica de tartas de protocolos. Tiene dependencia de la clase Main ya que instancia una ventana independiente del formulario.

- List: Lista formateada para datos para la gráfica
- Generar_Lista_Datos():Método privado para crear la lista de datos.
- Imprimir_Gráfica(): Método privado para imprimir por pantalla las gráfica.

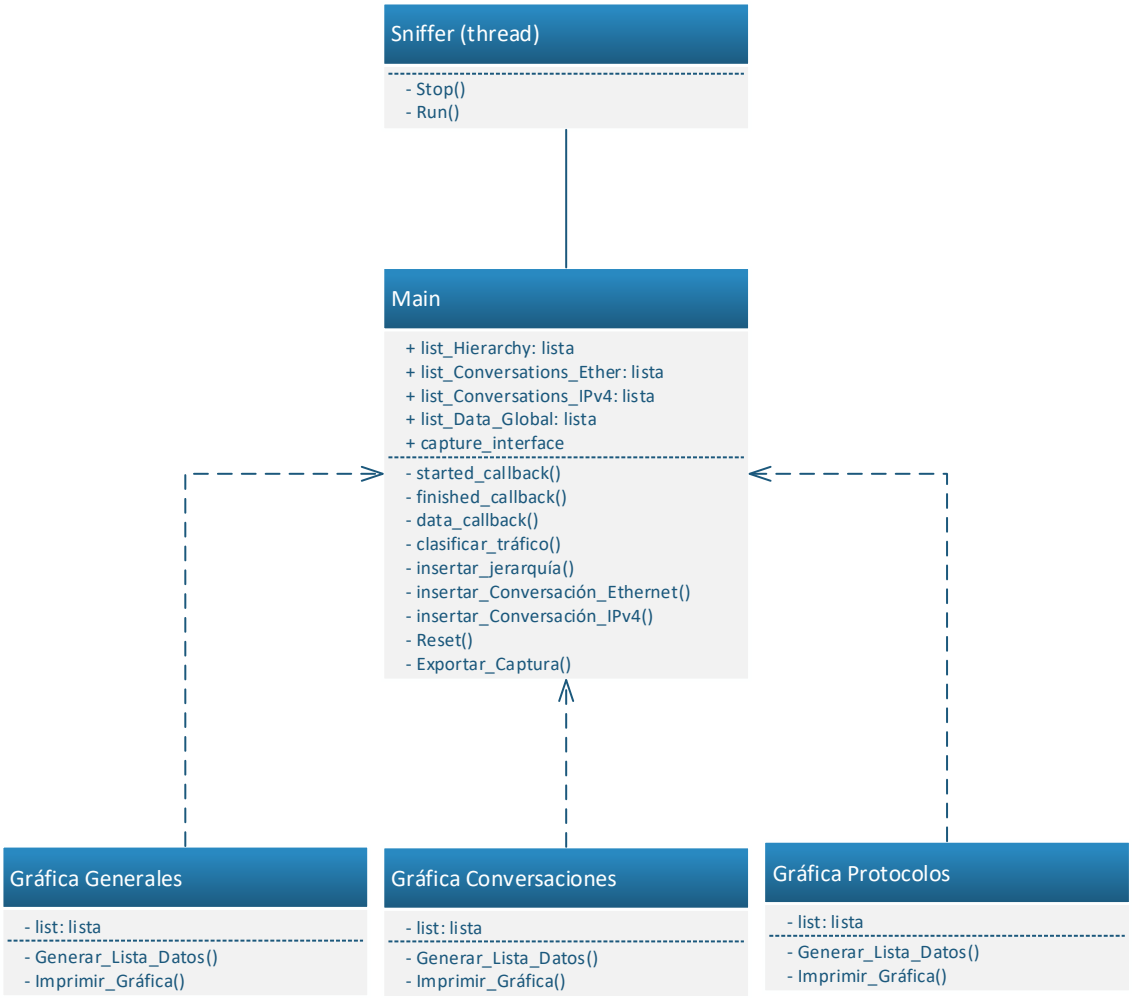


Ilustración 15. Diagrama de clases

3.4 Casos de uso

El diagrama de casos de uso presenta el comportamiento del programa en su iteración con el usuario. Como base, el usuario seleccionará la *interface* de red por donde se capturarán los datos, sin esta selección no se podrá activar la captura. El usuario podrá activar la captura, y en cualquier momento pude pausarla y reanudarla.

La clasificación y contabilización de datos funcionará mientras la captura esté activa. La jerarquía de protocolos y conversaciones *ethernet* e IPv4 realizarán sus procesos de forma automática una vez que el dato captura esté clasificado. El usuario en todo momento puede hacer consulta visual.

Por último, la visualización de estadísticas y gráficas estarán disponibles para que el usuario pueda consultarlas en todo momento.

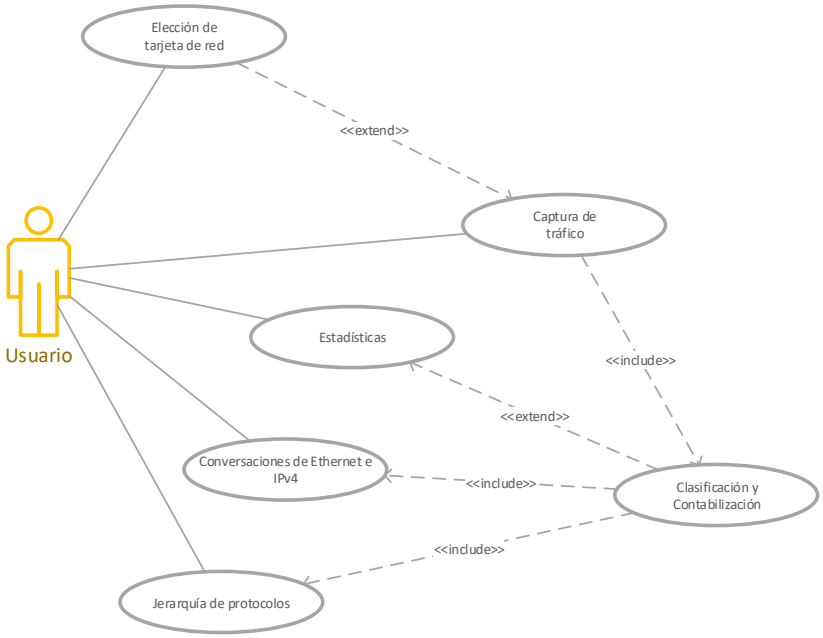


Ilustración 16. Diagrama de casos de uso

Caso 1	
Elección de tarjeta de red	
Descripción	Lo primero que se debe seleccionar es el dispositivo por donde se realizará la captura. El sistema deberá recopilar los *interfaces* de red disponibles, y mostrarlos en una lista.
Actores	Usuario
Casos de uso relacionado	Ninguno. Formulario inicial
Pre-Condiciones	Ninguna
Pos-Condiciones	Selección de una *interface* de red.
Flujo	Selección de tarjeta de red. Activación del modo captura.

Caso 2	
Captura de tráfico	
Descripción	Una vez que sepamos cual es la tarjeta fuente de la captura, obligaremos al usuario por medio de un botón, a iniciar la captura. En todo momento, el usuario tendrá la opción de detener la captura. La captura se realizará mediante hilos, por lo que conseguimos que mientras se está capturando, el programa puede continuar interactuando con el usuario.
Actores	Usuario
Casos de uso relacionado	Caso 1. Elección de tarjeta de red Caso 3. Clasificación de tráfico
Pre-Condiciones	Tener seleccionada una tarjeta de red
Pos-Condiciones	Activar el modo captura

Caso 3	
Clasificación y contabilización del tráfico	
Descripción	Se realizará un análisis de la trama capturada para su posterior contabilización. Esta clasificación estará vinculada al tipo de protocolo. Además, se clasificará por tipo de tráfico (*Unicast, Multicast y Broadcas*t). Además, se deberá contabilizar el tráfico (número de paquetes y tamaño en Bytes).
Actores	*Sniffer*
Casos de uso relacionado	Caso 2. Captura de tráfico
Pre-Condiciones	Captura de trama
Pos-Condiciones	Clasificar tráfico (*Unicast, Multicast, Broadcast*) (ARP, IP, ICMP, IGMP, TCP, UDP, HTTP, HTTPS. DNS…)
Alternativas y excepciones	Solo se clasificará tráfico *ethernet*

Caso 4	
Visualización de tráfico. Jerarquía de protocolos	
Descripción	Mientras el sistema está capturando tráfico, realizará una clasificación de los protocolos mediante jerarquía. Esta jerarquía se representará en formato árbol para una mejor visualización por parte del usuario.
Actores	*Sniffer* Usuario
Casos de uso relacionado	Caso 3. Clasificación de tráfico
Pre-Condiciones	Captura de tráfico
Pos-Condiciones	Jerarquía de protocolos

Caso 5	
Visualización de tráfico. Conversaciones Ethernet e IPv4	
Descripción	Mientras el sistema está capturando tráfico, este mostrará las conversaciones entre hosts, tanto a nivel *ethernet* como a nivel IPv4. En estas conversaciones se contabilizará el tráfico total como el sentido de este (de *Host* A -> *Host* B y viceversa).
Actores	*Sniffer* Usuario
Casos de uso relacionado	Caso 3. Clasificación de tráfico
Pre-Condiciones	Captura de tráfico
Pos-Condiciones	Conversaciones Ethernet e IPv4

Caso 6	
Visualización de estadísticas	
Descripción	Mientras el sistema está capturando tráfico, el usuario tendrá la opción de acceder a una donde podrá ver contadores del tráfico capturado. Además, tendrá la opción de crear gráficas para el análisis.
Actores	Usuario
Casos de uso relacionado	Caso 3. Clasificación de tráfico
Pre-Condiciones	Captura de tráfico
Pos-Condiciones	Estadísticas

3.5 GUI (Graphical User Interface)

Para la *interface* Gráfica de Usuario se ha pretendido separarlo en dos bloques. El primero sería la parte de visualización y contabilización de los datos. En este bloque se representaría la jerarquía de protocolos y las conversaciones *ethernet* e IPv4. En el segundo bloque estaría la parte de estadísticas y gráficas. Aquí, tendríamos unos contadores de tráfico generales y los botones o links para la generación de gráficas.

Habría un bloque fijo que sería la parte de captura. Este bloque siempre tendría que estar presente para usuario ya que en cualquier momento tendrá la posibilidad de detener y/o reanudar la captura.

En las siguientes imágenes representan un boceto de la GUI:

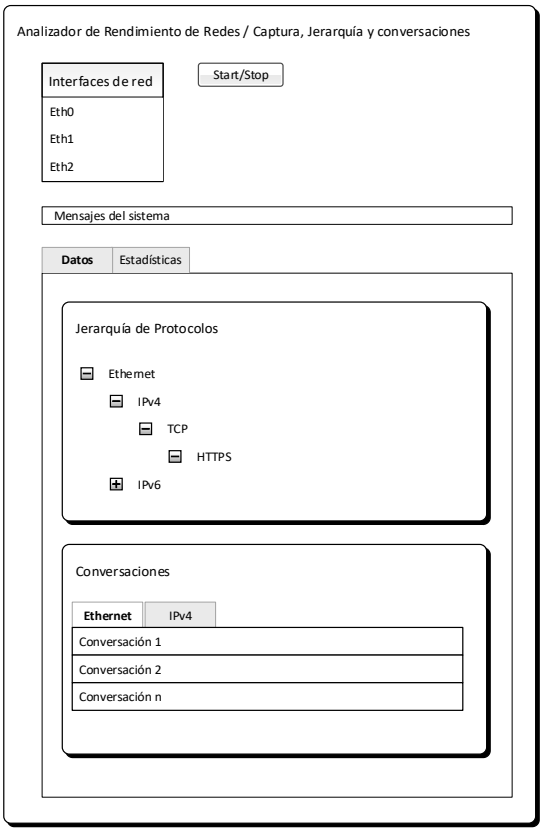


Ilustración 17. Boceto GUI captura y visualización

En la zona de las conversaciones tendremos disponibles un elemento tipo ficha para que el usuario puede seleccionar en cualquier momento que tipo de conversación quiere visualizar. El tipo de elemento ficha también lo utilizaremos para mostrar la zona de estadísticas.

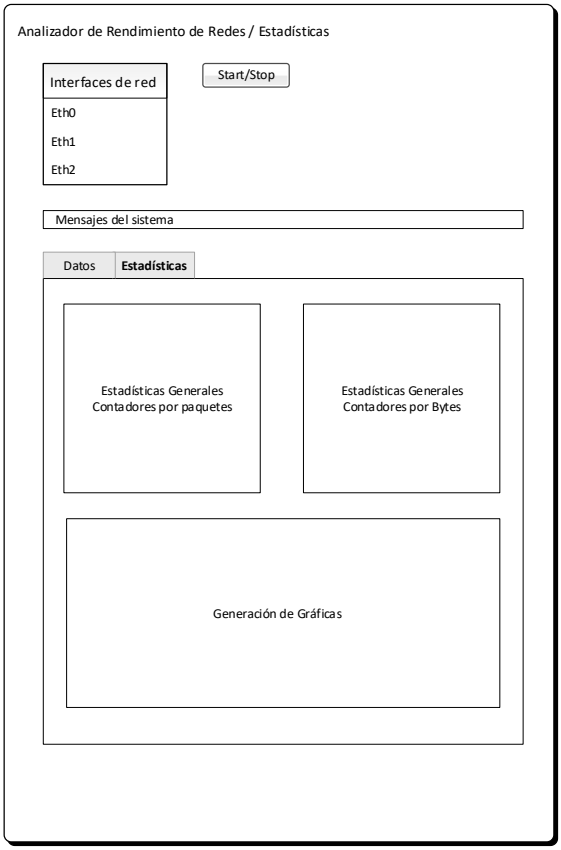


Ilustración 18. Boceto GUI Estadísticas

4. Implementación

En este capítulo se describen las fases de desarrollo de la herramienta. Se describirá el entorno de desarrollo y los desafíos que se han tenido que afrontar para su definición.

4.1 Entorno de desarrollo

Para la implementación de la herramienta he elegido el lenguaje de programación Python. Este es un lenguaje de programación interpretado, multiparadigma y de programación imperativa. Además, Python tiene licencia de código abierto con licencia pública general GNU (*General Public License*).

Las premisas para la elección del lenguaje de programación eran que tenía que ser de fácil uso, código abierto, programación gráfica basada en formularios, multiplataforma y documentación amplia y de fácil acceso.

Se valoraron otros lenguajes como Java, C o C# que, aunque también ofrecían desarrollo multiplataforma, yo no poseía la experiencia necesaria como para afrontar un proyecto de este calibre.

Para la parte de la captura de datos se ha utilizado la librería Scapy. Es una herramienta de gestión de paquetes de que utiliza la librería *libpcap* (*WinPCap/Npcap* en Windows). Es una herramienta que es muy utilizada en el entorno del hacking y además es utilizada por Wireshark.

En la parte de diseño gráfico se ha utilizado la herramienta Qt Designer para el diseño de ventanas y formularios. Esta herramienta permite generar de forma sencilla formularios y transformarlos a código.

Para la parte de gráficas estadísticas se ha utilizado la librería *matplotlib*. Es una biblioteca para la generación de gráficos a partir de datos contenidos en listas o *arrays* en Python.

El entorno de desarrollo es el siguiente:

- Ordenador: MacBook Pro 2010 (Intel i5 2,4 GHz 8GB memoria)
- Sistema Operativo: macOS High Sierra (10.13.6)
- IDE (*Integrated Development Environment*): PyCharm ver.2019.2.3
- Python 3.7

Las versiones de las librerías utilizadas son las siguientes:

- PyQt5 5.13.1
- Matplotlib 3.1.2

- Netifaces 0.10.9
- Psutil 5.6.7
- Scapy 2.4.3

4.2 Interfaces de red

Como en todo analizador de redes se tiene la opción de seleccionar la *interface* de red por donde se quiere capturar los datos. Se ha implementado el elemento de formulario tipo *combobox*, donde previamente se le rellena con todos los *interfaces* de red disponibles en el sistema donde se ejecuta la herramienta.

El usuario tiene la opción de elegir la *interface* por donde quiere realizar la captura. La herramienta solo puede capturar tráfico por una única tarjeta de red.

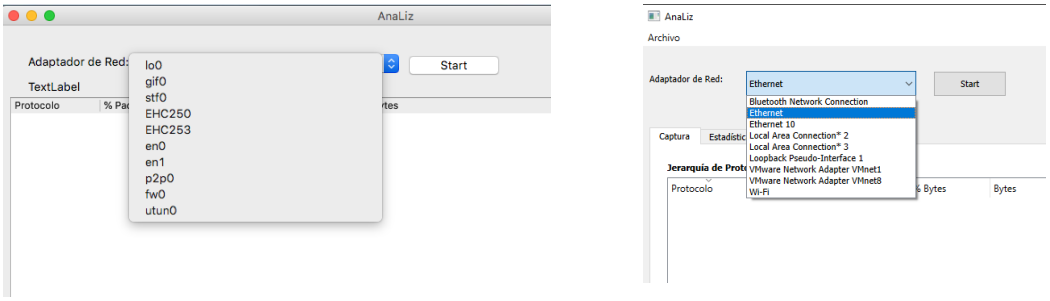


Ilustración 19. Captura del combobox con las interfaces detectadas tanto en macOS y Windows

Dependiendo del sistema operativo donde se ejecute la aplicación, se debe tener en consideraron las librerías que se deben importar. En el caso de Linux y macOS, la librería a usar es *netifaces*, pero en sistemas Microsoft Windows esta es *psuitl*. Con el siguiente código solucionamos el problema de en qué plataforma se ejecuta.

```
# Obtenemos todos los interfaces disponibles de la máquina dependiendo del SO
if _platform == "win32" or _platform == "win64":  # Si el SO es MS Windows
    import psutil
    list_Interfaces = []
    addres = psutil.net_if_addrs()
    for i in sorted(psutil.net_if_addrs().keys()):
        list_Interfaces.append(i)
else: #si el SO macOS o Linux
    import netifaces
    list_Interfaces = netifaces.interfaces()
```

4.3 Captura de tráfico

Una de las partes más importantes de la herramienta es la captura de datos, porque evidentemente, si no hay datos no se puede analizar nada. En la primera fase de implementación se realizó la parte de captura, y su funcionamiento era correcto. El problema se manifestó cuando se quería capturar y clasificar tráfico al mismo tiempo, ya que se detectaba que había pérdida de información.

Para solventar el problema se recurrió a una solución de programación *multithread*. La herramienta tiene dos procesos que se ejecutan en paralelo, uno con la captura y otro con la clasificación y visualización.

Se barajó la posibilidad de crear otro *thread* para la contabilización del tráfico, pero se descartó porque este proceso no incrementaba el rendimiento.

En el siguiente diagrama de flujo se explica la ejecución del proceso de captura. Al ser una *thread*, este se ejecutará en modo *loop*, mientras que el usuario no detenga la captura. La sincronización del proceso se realiza mediante *mutex.*

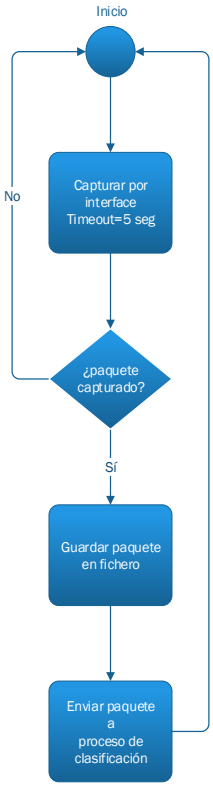


Ilustración 20. Diagrama de flujo thread de captura de datos

Sniffer

Scapy es una utilidad de Python que permite al usuario enviar, capturar, diseccionar y cambiar paquetes de red. Permite a los programadores realizar potentes herramientas para escáner o atacar redes.

Para el desarrollo de del analizador se ha utilizado el comando *sniff*. Este comando ofrece una serie de parámetros para configurar la captura (https://scapy.readthedocs.io/en/latest/api/scapy.sendrecv.html). En nuestro caso solo necesitamos tres: la *interface* por donde se captura, el número de paquetes que captura y el *timeout*. La instrucción que ejecutamos es la siguiente:

```
# captura un único paquete con un timeout de 5 segundos
packet_sniffed = sniff(iface=capture_interface, count=1, timeout=5)
```

Sniff nos devuelve una lista con los paquetes capturados. Por ejemplo, si el parámetro *count* lo hubiésemos establecido en 3, nos devolvería una lista de 3 elementos, y para acceder al último paquete lo haríamos con el índice [2].

Como estamos en un proceso repetitivo (*multithread*), únicamente nos interesa capturar un único paquete, para su posterior envío al proceso de clasificación. Además, nos apoyamos en el parámetro *timeout* para controlar la falta de tráfico por cualquier motivo. En esta configuración el *timeout* está establecido en 5 segundos, así aseguramos que no se produzca error en la captura.

4.4 Exportación de datos

En el punto de requisitos de la herramienta de la fase de diseño, se apunto que la exportación de datos a un fichero *.pcap* era una mejora. En el desarrollo de la aplicación y más concretamente en la fase de pruebas, se ha visto necesario aplicar esta mejora.

Scapy tiene una utilidad para el tratamiento de ficheros, es decir oferta herramientas para la lectura y escritura de archivos en formato *pcap*. Cada vez que se captura un paquete, la aplicación guarda la información mediante el comando *wrpcap*. En este caso utilizamos el parámetro *append* para añadir el dato al fichero. Si este fichero no existiera, *wrpcap* lo crea.

```
# Guarda el paquete capturado en un fichero temporal
wrpcap(capture_file, packet_sniffed, append=True)
```

Mientras se está capturando se crea un fichero temporal donde se guardan los datos. Este fichero es borrado cuando el usuario sale de la aplicación, o cuando el usuario quiere empezar una captura nueva. En todo momento se puede guardar en disco la captura.

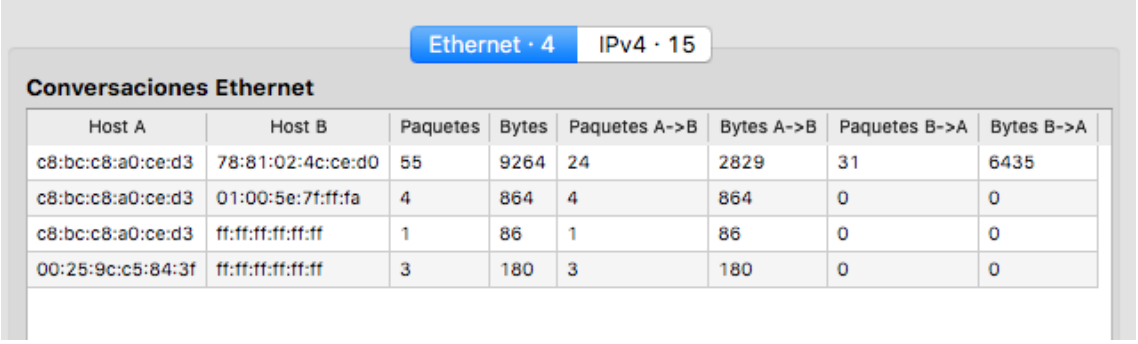


Ilustración 21. Conversaciones Ethernet

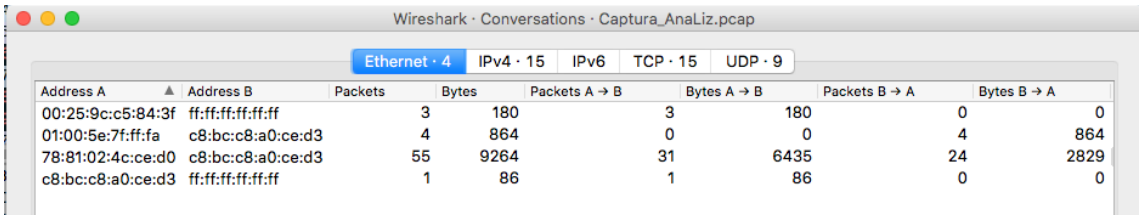


Ilustración 22. Lectura de fichero de captura abierto en Wireshark

La exportación de datos permite comparar la captura y los resultados con otros analizadores del mercado. Esto ha facultado la corrección de errores tanto en la captura como en la contabilización de datos para las estadísticas.

4.5 Clasificación del tráfico

En la parte de clasificación es donde más retos se han planteado con sus soluciones. La decisión de visualizar el tráfico con una jerarquía de protocolos ha sido en la que más tiempo se ha empleado. En un principio únicamente se iban a visualizar los protocolos más relevantes de cada capa, con lo que se quedaba en unos 10 protocolos como máximo. Con esta solución el alcance de clasificación de protocolos es mucho mayor, ya que ahora podemos reconocer unos 350

Para ello, se han creado listas de protocolos y se han indexado por el número que asigna IANA (https://www.iana.org/). Se ha desarrollado una función donde se pasa por parámetro este número y retorna el nombre del protocolo. Si este no está en la lista devolverá "*unknown*". Como ejemplo, el siguiente código describe la función que devuelve el nombre de protocolo en la capa de Internet:

```
def GetType_Ether(self,num_protocol):
    list_pro = {2048: 'IPv4', 2054: 'ARP', 2114: 'WakeOnLAN', 8947:'TRILL',
8938: 'SRP', 24579: 'DECNET', 32821: 'RARP', 32923: 'APPLETALK', 33011: 'AARP',
33024: 'VLAN',33079: 'IPX', 33284: 'QNX', 34525: 'IPv6', 34824: 'EFC', 34825:
'ESP', 34841: 'COBRANET', 34887: 'MPLS-UNICAST',
34888:  'MPLS-MULTICAST',  34915:  'PPPOE-DIS',  34916:  'PPPOE-SES',  34925:
'INTEL',  34928:  'JUMBO-FRAME',  34939:  'HOMEPLUG',  34958:'EAP',  34962:
'PROFINET', 34970: 'HYPERSCSI', 34978: 'AOE', 34980: 'ETHERCAT',
 35020: 'LLDP'}

    if not num_protocol in list_pro:
        return "unknown"
    else:
        return list_pro[num_protocol]
```

Hay una función de retorno de protocolo por cada capa del modelo TCP/IP.

En el siguiente diagrama de flujo se describe el proceso que se sigue para la clasificación del tráfico. Aquí se puede apreciar que el tráfico no *ethernet* se descarta.

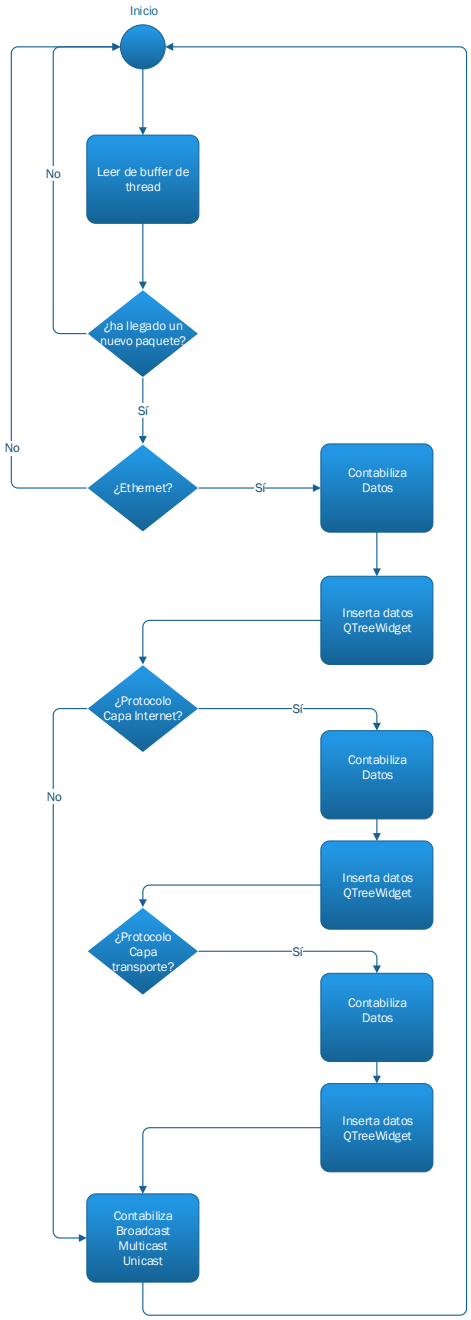


Ilustración 23. Diagrama de flujo para la clasificación del tráfico

Librería Scapy

Para poder clasificar el paquete capturado necesitamos saber la información que contiene este. La opción más evidente es hacerlo a bajo nivel, por lo que habría que transformar la información del paquete en hexadecimal o binario, y por las posiciones de los bits accederíamos a las cabeceras Ethernet e IP.

Con la librería Scapy esta operación se simplifica enormemente, ya que nos permite operar con los paquetes capturados para obtener información. Aquí, estamos es disposición de obtener toda la información de las capas, como direcciones MAC, direcciones IP, tipo de protocolo, etiquetas, puertos, etc. En la

Ilustración 23 describe la relación de los atributos con la posición de bits de un paquete capturado:

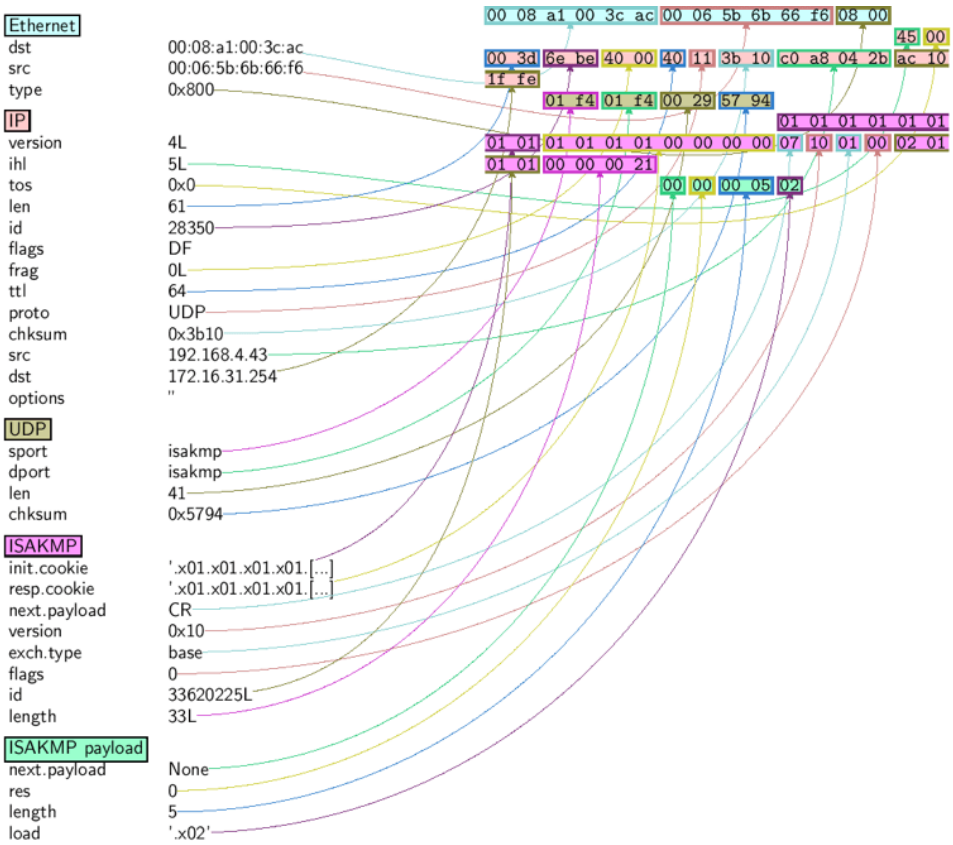


Ilustración 24. Correspondencia de atributos de la librería Scapy

Por ejemplo, si queremos obtener la dirección IP destino de un paquete capturado escribiríamos el siguiente comando (Python Console):

```
>>>data_packet(IP).dst

>>> 172.16.31.254
```

Gracias a esta librería se facilitó la clasificación del tráfico, pero quedaba la parte de obtener el tamaño del paquete. Consultado la documentación, el atributo *.len*, nos devuelve el tamaño de la capa correspondiente, pero no de las otras capas que conforma el paquete, por lo que se tendría que hacer una suma de todas ellas. Además, con otros protocolos como ARP el atributo de tamaño es *.plen*.

La solución es muy simple, únicamente hay que calcular el tamaño del dato capturado con el comando *len* de Python, y nos devolverá un número que coincide con el tamaño en Bytes.

```
>>>len(data_packet[0])

>>> 79
```

https://stackoverflow.com/questions/38886709/python-scapy-packet-size-difference

4.6 Jerarquía de protocolos

Una vez clasificado el paquete capturado se necesita presentarlo. En la parte de conceptualización se analizaron varias herramientas que existen el mercado, y la mejor aplicación que presentaba los datos es Wireshark con la jerarquía de protocolos (Ilustración 10).

La jerarquía de protocolo se basa en presentar la información de red en un formato de árbol, por lo que a medida que se van abriendo las ramas, se despliegan los subárboles a los que pertenecen. Esto brinda al usuario en una simple vista una esquematización de la información muy intuitiva y fácil de comprender.

Desarrollar esta jerarquía de protocolos en este proyecto se optó por el uso del *widget* de la librería PyQt5 QTreeWidget. Este *widget* se compone de objetos los cuales contienen información del ítem (columnas, hijos, parientes, etc.).

Paralelamente a la inserción de ítems en el *widget*, se construye una estructura de datos que se compone una tabla que en Python se denomina diccionario. En este diccionario tomamos como índice la concatenación del nombre de los protocolos y el objeto ítem del *widget*. Los datos quedarían almacenados de la siguiente manera:

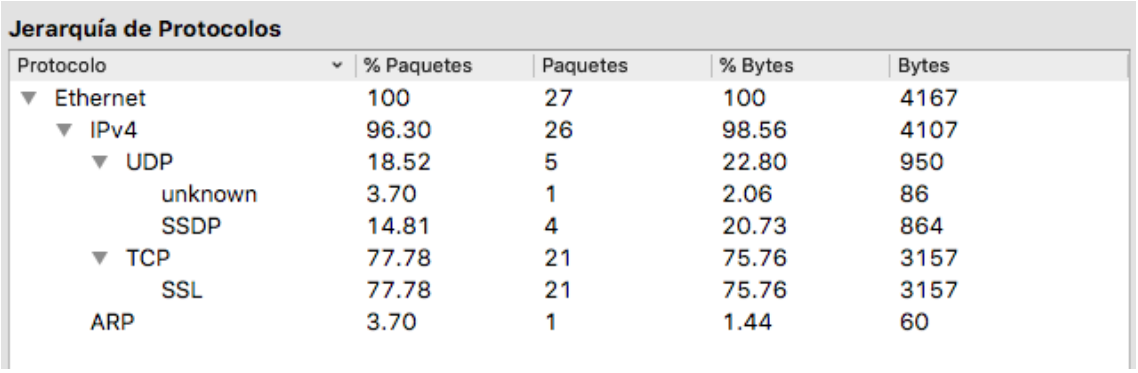
{'IPv4': <PyQt5.QtWidgets.QTreeWidgetItem object at 0x11fff2a68>,
'IPv4_TCP': <PyQt5.QtWidgets.QTreeWidgetItem object at 0x11fff2af8>,
'IPv4_TCP_SSL': <PyQt5.QtWidgets.QTreeWidgetItem object at 0x11fff2b88>,
'ARP': <PyQt5.QtWidgets.QTreeWidgetItem object at 0x1200205e8>}

Los índices están representados de color verde, y el contenido de color naranja. De esta manera en cualquier momento podemos acceder a la información del ítem para consumir o actualizar datos.

Contabilización de datos

Como siguiente paso a la forma de guardar la información se necesita estructurar los datos que se quieren contabilizar. Para el análisis y posterior representación de estadísticas he ha decido contabilizar número de paquetes y tamaño de estos en Bytes.

Por consecuente, los datos que guardaremos en el ítem de la estructura de datos serán número paquetes, Bytes y el cálculo del porcentaje que representan estos respecto a su número total en la captura. Para ilustrarlo, tomaremos como ejemplo la Ilustración 24 donde hay una captura de datos con su correspondiente clasificación:



Jerarquía de Protocolos

Protocolo	% Paquetes	Paquetes	% Bytes	Bytes
▼ Ethernet	100	27	100	4167
▼ IPv4	96.30	26	98.56	4107
▼ UDP	18.52	5	22.80	950
unknown	3.70	1	2.06	86
SSDP	14.81	4	20.73	864
▼ TCP	77.78	21	75.76	3157
SSL	77.78	21	75.76	3157
ARP	3.70	1	1.44	60

Ilustración 25. Ejemplo de jerarquía de protocolos

En la fila donde se ubica el protocolo SSDP (*Simple Service Discovery Protocol*) vemos que se han capturado 4 paquetes con un total de 864 Bytes, por lo representa el 14,81% de paquetes capturados y el 20,73% de Bytes procesados.

Subiendo en la jerarquía observamos que la suma de paquetes y de Bytes del protocolo UDP es exactamente la suma de los protocolos *unknown* y SSDP, por tanto, UDP representa el 18,52% del total de paquetes capturados y el 22,80% de Bytes procesados.

4.7 Conversaciones entre hosts

Otra forma que tenemos de analizar el rendimiento de nuestro tráfico se saber qué conversaciones está habiendo en la red. Básicamente en esta clasificación consta de un host A que se comunica con un host B y analizar sus datos, así como la dirección de estos.

En este proyecto se analizarán las conversaciones ethernet e IPv4, ya que se consideran las más frecuentes e importantes. Las conversaciones ethernet se basan en las direcciones MAC origen y destino y para las conversaciones IPv4 las direcciones IP origen y destino.

Es importante remarcar que en una red el número de conversaciones *ethernet* e IPv4 difieren es su número porque, por ejemplo, un ordenador puede estar conversando con un *router* y esto sería una única conversación (*ethernet*), pero la misma máquina puede estar conversando con varios ordenadores que esté por detrás del *router* y estas conversaciones son IPv4.

Para representar las conversaciones se utilizó el *widget* de la librería PyQt5 QTableWidget. Este básicamente se compone de ítems que representan filas y estas filas están formadas por columnas.

Como en la jerarquía de protocolos, paralelamente a la inserción de filas se fabricará una estructura de datos con un diccionario. El índice será la concatenación de la MAC origen y destino para conversaciones ethernet, y la IP origen y destino para las conversaciones IPv4. Los datos quedarán almacenados de la siguiente manera:

{'c8:bc:c8:a0:ce:d3_78:81:02:4c:ce:d0': [0, 15, 2516, 9, 964, 6, 1552],
'c8:bc:c8:a0:ce:d3_01:00:5e:00:00:fb': [1, 1, 87, 1, 87, 0, 0],
'c8:bc:c8:a0:ce:d3_01:00:5e:7f:ff:fa': [2, 1, 168, 1, 168, 0, 0]}

Observamos que el elemento del diccionario es una lista de siete elementos.

- [0]: Número de fila del *widget.*
- [1]: Número de paquetes de la conversación.
- [2]: Tamaño en Bytes de la conversación.
- [3]: Número de paquetes en sentido A->B.
- [4]: Tamaño en Bytes en sentido A->B.
- [5]: Número de paquetes en sentido B->A.
- [6]: Tamaño en Bytes en sentido B->A.

Contabilización de datos

Se define como una conversación el paquete capturado que tiene como origen A y va dirigido a un destino B, y además se considerará la misma conversación si otro paquete capturado tiene como origen B y destino A. Por consiguiente, se contabilizará como una misma conversación y se actualizarán en su correspondiente fila.

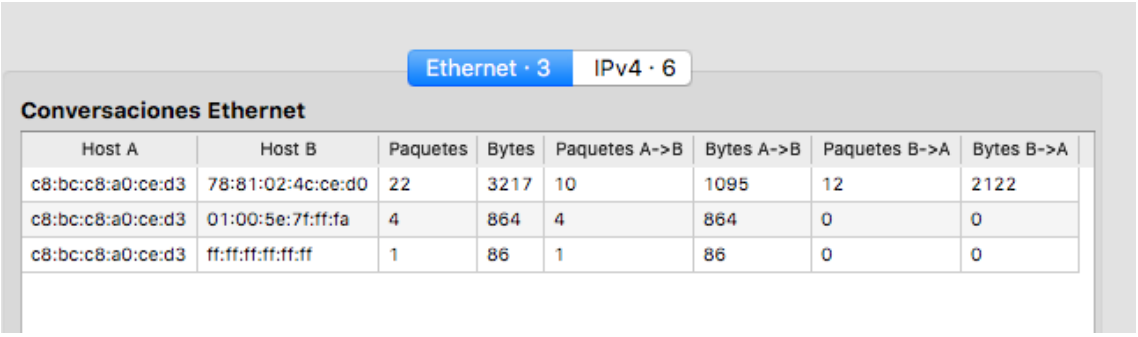
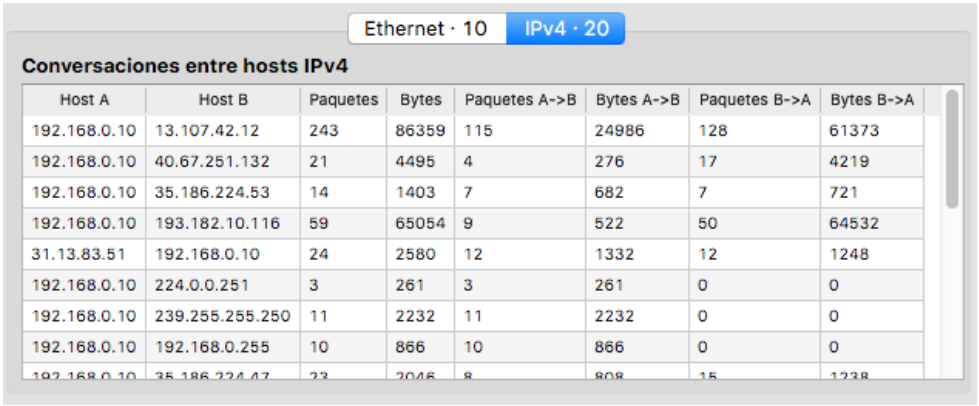


Ilustración 26. Ejemplo de conversaciones Ethernet

Observamos que en la ficha de conversaciones ethernet como en la de IPv4, está incluido el número total de conversaciones respectivamente.



Host A	Host B	Paquetes	Bytes	Paquetes A->B	Bytes A->B	Paquetes B->A	Bytes B->A
192.168.0.10	13.107.42.12	243	86359	115	24986	128	61373
192.168.0.10	40.67.251.132	21	4495	4	276	17	4219
192.168.0.10	35.186.224.53	14	1403	7	682	7	721
192.168.0.10	193.182.10.116	59	65054	9	522	50	64532
31.13.83.51	192.168.0.10	24	2580	12	1332	12	1248
192.168.0.10	224.0.0.251	3	261	3	261	0	0
192.168.0.10	239.255.255.250	11	2232	11	2232	0	0
192.168.0.10	192.168.0.255	10	866	10	866	0	0
192.168.0.10	35.186.224.47	23	2046	8	808	15	1238

Ilustración 27. Ejemplo de conversaciones IPv4

4.8 Estadísticas y gráficas

La fase de estadísticas es sin duda la parte más importante del proyecto, ya que aquí es donde refleja todo el trabajo descrito en los puntos anteriores. Además, es los que el usuario utilizará para comprobar el estado de la red y sacar sus propias conclusiones.

Cada vez que se ha contabilizado un paquete válido, se han ido construyendo una estructura de datos donde se guardaban los datos globales del tráfico. Esta estructura es una “lista de listas” donde capa posición de la lista es un contador global estadístico. Un ejemplo de esta estructura es el siguiente:

<class 'list'>: [[25, 3391], [4, 266], [6, 491], [5, 326], [0, 0], [20, 3065]]

La lista se compone de 6 elementos donde:

- [0]: Total de tráfico analizado.
- [1]: Conversaciones *Ethernet*.
- [2]: Conversaciones IPv4.
- [3]: Tráfico *Broadcast*.
- [4]: Tráfico *Multicast*.
- [5]: Tráfico *Unicast*.

Además, cada elemento de la lista se compone de una lista de 2 elementos, donde el primero representa al número de paquetes y el segundo al tamaño en Bytes. En la Ilustración 27 se muestra cómo se representan los contadores globales de estadistas en la aplicación.

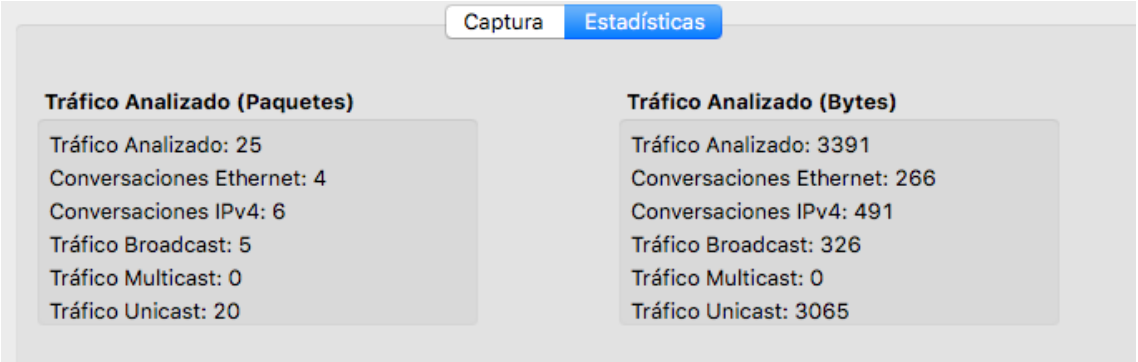


Ilustración 28. Totales tráfico analizado

Gráficas

A la hora de afrontar la parte gráfica se ha querido darle un aire de personalización para la representación visual. El usuario tiene la opción de generar su propia gráfica respecto a lo que quiere analizar, en relación con la información dispone.

En la parte gráfica el usuario dispone de opciones para generar gráficas por paquetes, por conversaciones y por tipo de protocolo. Por ejemplo, si queremos la gráfica de conversaciones IPv4 por Bytes, seleccionaremos estas opciones y como resultado nos saldrá un gráfico de barras como el de la Ilustración 28.

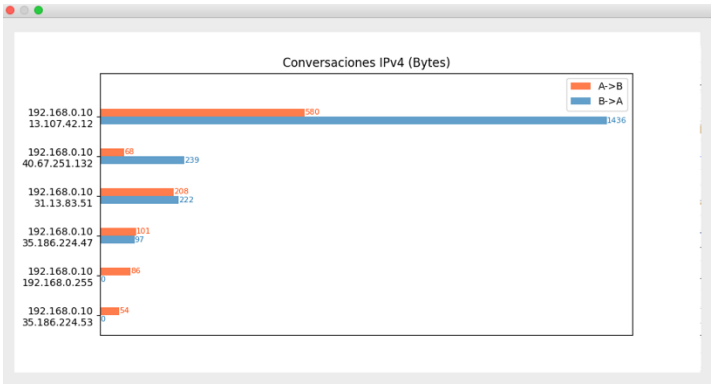


Ilustración 29. Gráfico conversaciones IPv4 por Bytes

En la parte de protocolos, se ha planteado el mismo sistema, pero con la salvedad que el los *combobox* únicamente saldrán protocolos que existan en la captura. Además, estas gráficas serán de formato tarta.

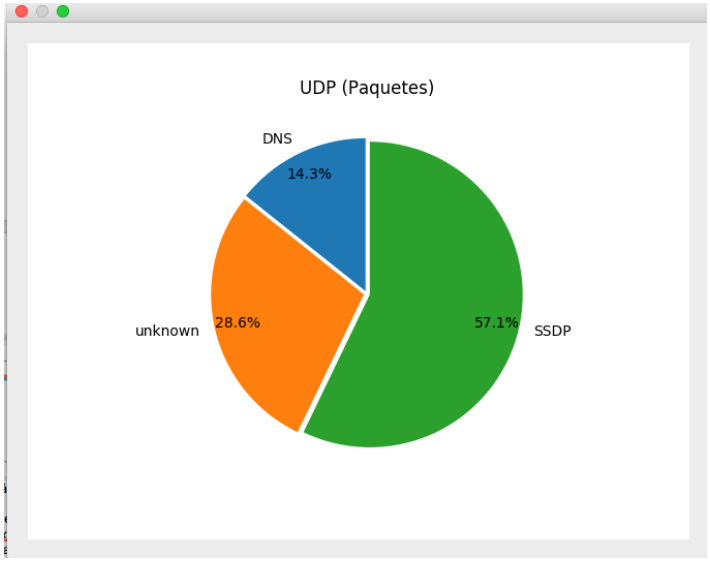


Ilustración 30. Gráfico tráfico UDP por número de paquetes

Cuando en la captura hay un único tipo de protocolo la gráfica de tipo tarta mostrará el 100%. Esta representación no ofrece mucha información, y para resolverlo se introduce el total de paquetes analizas o tamaño en Bytes, según la elección del usuario. Esto creará una gráfica comparando el tráfico del protocolo respecto al tráfico total.

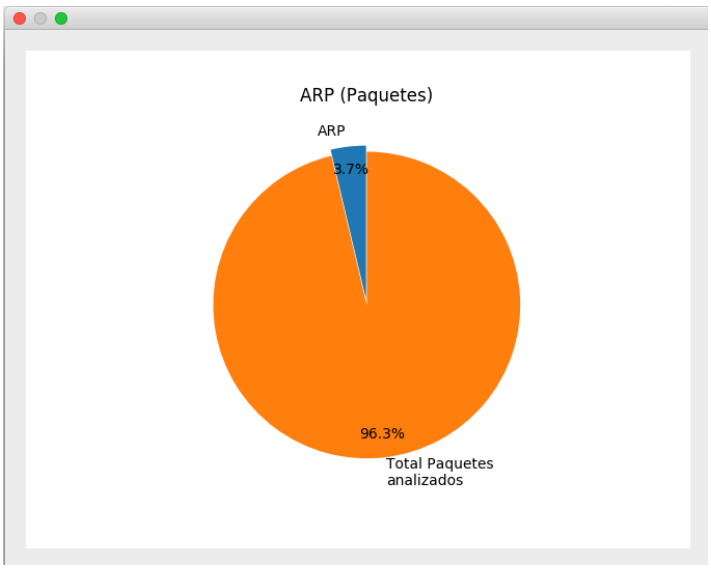


Ilustración 31. Gráfica ARP respecto al total de paquetes analizados

Por último, también está la opción de visualizar una foto en modo gráficas del todo el tráfico capturado. Esta pantalla está compuesta de seis gráficas de formato tarta donde se representa el tipo de tráfico, tráfico Ethernet, tráfico IPv4, tráfico IPv6, tráfico TCP y tráfico UDP.

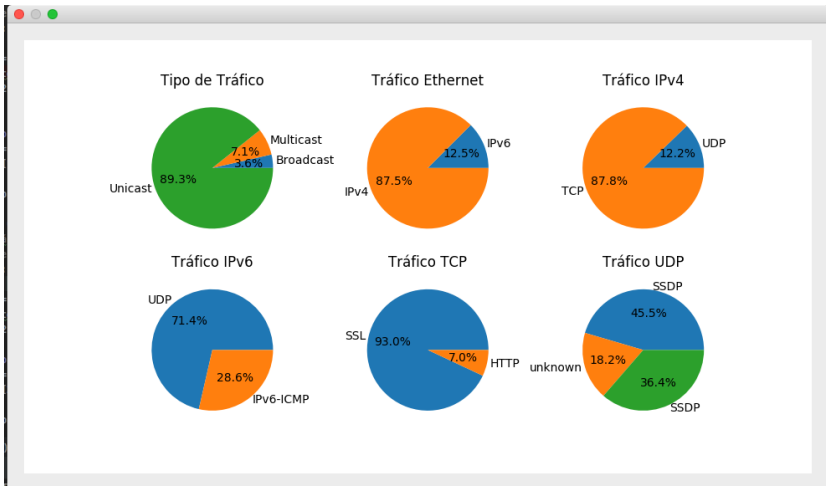


Ilustración 32. Gráficas de tráfico General

Para la visualización de las gráficas se ha optado por abrir una ventana nueva y mostrar la información. De esta forma el usuario tiene la opción de mover la pantalla y continuar visualizando la captura y de maximizar la venta de gráficas para tener mejor vista de ella. Un problema que no he podido resolver es poder continuar manipulando la ventana de la aplicación mientras que la gráfica este abierta, por lo que obligatoriamente hay que cerrar la ventana de la gráfica.

Otro problema que no he podido resolver es el “*overlabel*”. Cuando tenemos una gráfica con muchas etiquetas, estas empiezan a montarse unos encima de otras por lo que la lectura de la gráfica se hace difícil.

5. Manual de usuario

La aplicación se distribuye para la plataforma Windows. El zip entregado del TFG contiene una carpeta que se llama “software”. Dentro de esta tenemos la aplicación en la carpeta “AnaLiz” y el software necesario para su funcionamiento (WinPcap).

Microsoft Windows

Para la ejecución en el sistema Microsoft Windows se necesita tener instalado previamente WinPcap. Se dispone de la última versión (WinPcap_4_1_3.exe) en la carpeta “software” para su instalación.

Una vez que WinPcap esté instalado se tendrá que abrir la carpeta “AnaLiz” y ejecutar el programa AnaLiz.exe. Se recomienda que se tengan permisos de ejecución como administrador.

Ejecución

Como vemos en la Ilustración 32 la aplicación tiene 3 zonas de trabajo:

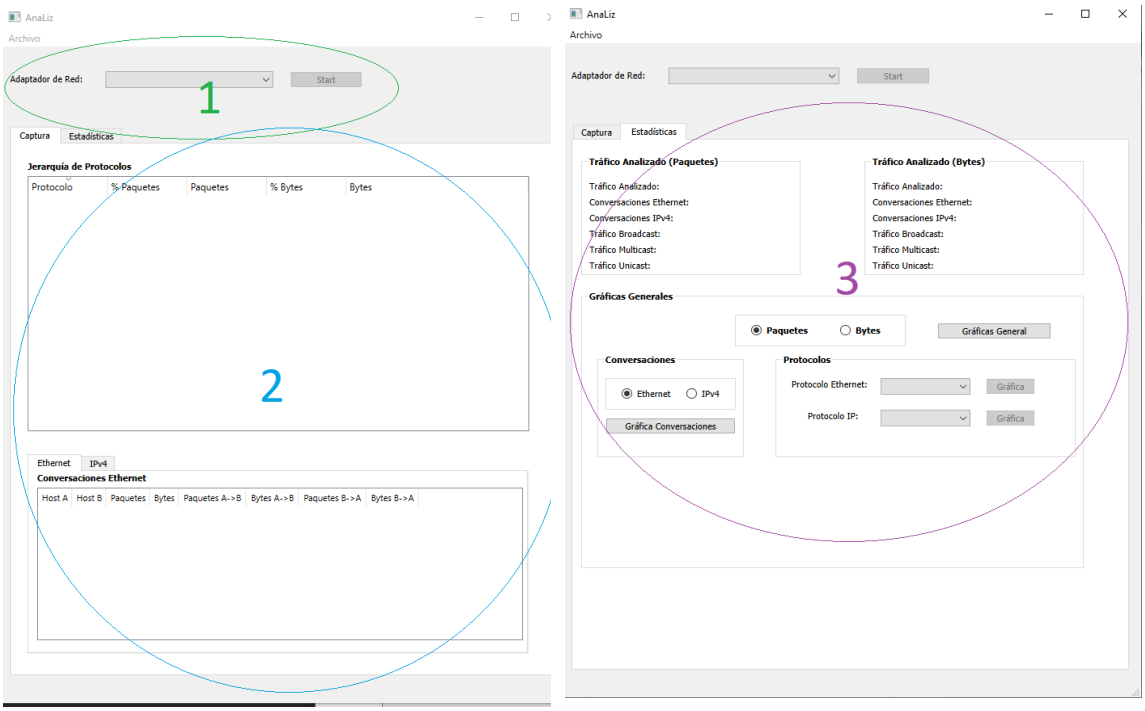


Ilustración 33. Zonas de trabajo de la aplicación

- Zona 1 (verde): Zona de captura de datos.
- Zona 2 (Azul): Zona de visualización de tráfico capturado.
- Zona 3 (Morado): Zona de estadísticas.

Para pasar de la zona 2 a la zona 3, se podrá realizar mediante las pestañas que están situadas encima de las zonas, indicando si es captura o estadísticas. La zona 1 siempre estará visible.

Captura de datos

Para la captura de datos siempre se trabajará en la zona 1. Lo primero que hay que hacer es seleccionar una *interface* de red por donde queremos realizar la captura (Ilustración 33). Si desplegamos en menú de “Adaptadores de red”, tendremos a disposición todos los interfaces de red que tengamos disponibles en el sistema.

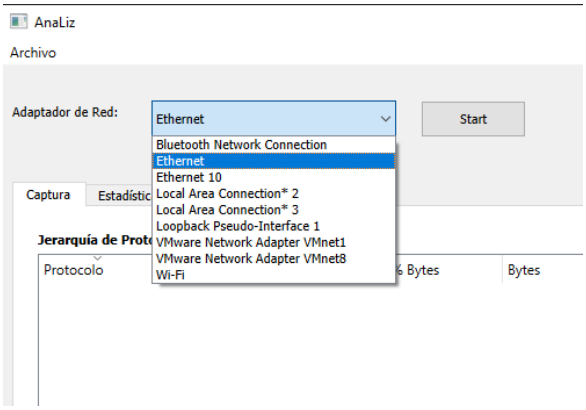


Ilustración 34. Elección de adaptador de red

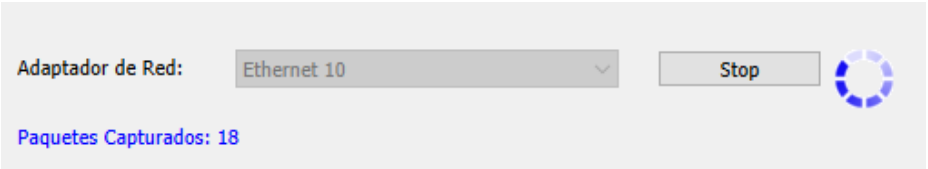


Ilustración 35. Capturando datos

Cuando tengamos seleccionada la tarjeta de red el botón de START queda habilitad y si pulsamos el sistema comienza a capturar datos. En este momento saldrá una rueda en movimiento que nos indicará que el sistema está trabajando (Ilustración 34). Además, nos saldrá un mensaje en azul indicando el número de paquetes capturados.

Si por alguna razón el sistema está trabajando (ya que la rueda azul está en movimiento) pero no está capturando, este nos avisa con el mensaje en rojo: “La *interface* no recibe datos… espere un momento” (Ilustración 35).

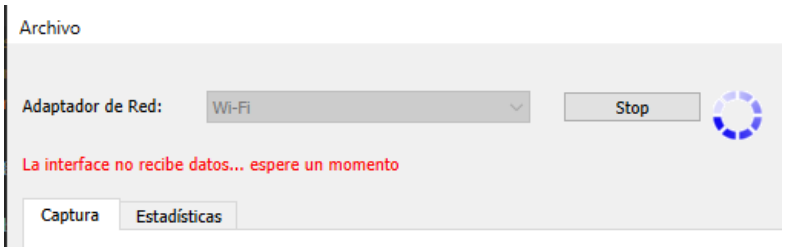


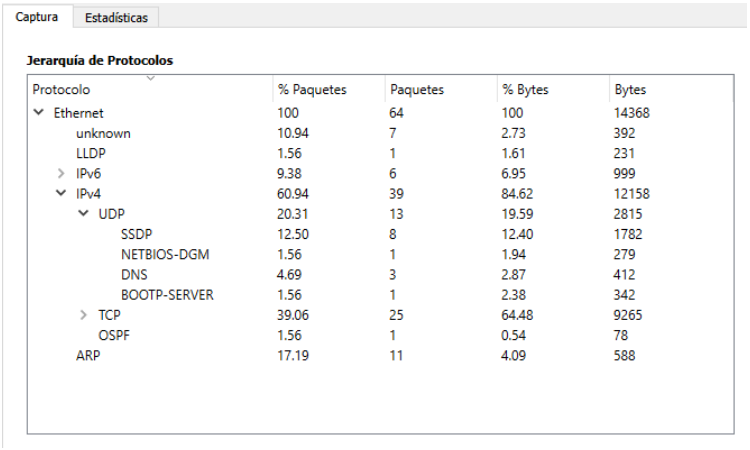
Ilustración 36. El sistema no está capturando datos

En cualquier momento el usuario puede pulsar STOP para pausar la captura. En este momento tiene la posibilidad de cambiar de *interface* de red.

Visualización de datos

En la zona 2 tenemos la parte donde se visualizan los datos capturados. Este tiene dos partes: la jerarquía de protocolos y las conversaciones Ethernet e IPv4.

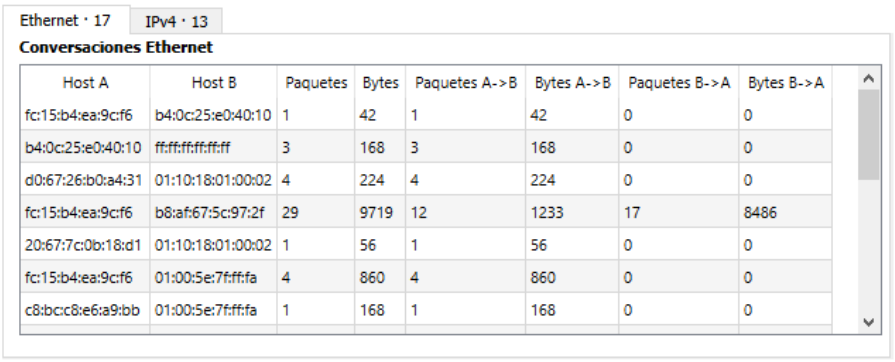
En la jerarquía de protocolos los paquetes que captura el sistema los ordena atendiendo el modelo TCP/IP en forma de árbol. Los datos que visualiza son los paquetes y el tamaño en Bytes y además muestra los porcentajes respecto a la jerarquía. El usuario puede extender o enrollar los diferentes subárboles a su interés.



Protocolo	% Paquetes	Paquetes	% Bytes	Bytes
Ethernet	100	64	100	14368
unknown	10.94	7	2.73	392
LLDP	1.56	1	1.61	231
IPv6	9.38	6	6.95	999
IPv4	60.94	39	84.62	12158
UDP	20.31	13	19.59	2815
SSDP	12.50	8	12.40	1782
NETBIOS-DGM	1.56	1	1.94	279
DNS	4.69	3	2.87	412
BOOTP-SERVER	1.56	1	2.38	342
TCP	39.06	25	64.48	9265
OSPF	1.56	1	0.54	78
ARP	17.19	11	4.09	588

Ilustración 37. Despliegue de información en la jerarquía

Para visualizar las conversaciones pasar de Ethernet a IPv4 lo haremos pulsando la ficha correspondiente.


Ethernet · 17 IPv4 · 13

Conversaciones Ethernet

Host A	Host B	Paquetes	Bytes	Paquetes A->B	Bytes A->B	Paquetes B->A	Bytes B->A
fc:15:b4:ea:9c:f6	b4:0c:25:e0:40:10	1	42	1	42	0	0
b4:0c:25:e0:40:10	ff:ff:ff:ff:ff:ff	3	168	3	168	0	0
d0:67:26:b0:a4:31	01:10:18:01:00:02	4	224	4	224	0	0
fc:15:b4:ea:9c:f6	b8:af:67:5c:97:2f	29	9719	12	1233	17	8486
20:67:7c:0b:18:d1	01:10:18:01:00:02	1	56	1	56	0	0
fc:15:b4:ea:9c:f6	01:00:5e:7f:ff:fa	4	860	4	860	0	0
c8:bc:c8:e6:a9:bb	01:00:5e:7f:ff:fa	1	168	1	168	0	0

Ilustración 38. Conversaciones Ethernet e IPv4

Estadísticas

La zona de estadísticas tiene dos partes, en la primera están ubicados los contadores generales sobre la captura, clasificación y contabilización del sistema. La segunda parte es la relativa a las gráficas. En esta última, el usuario tiene la posibilidad de visualizar unas gráficas generales o personalizar su propio gráfico.

Para mostrar las gráficas generales únicamente hay que pulse “Gráficas Generales (Ilustración 38).

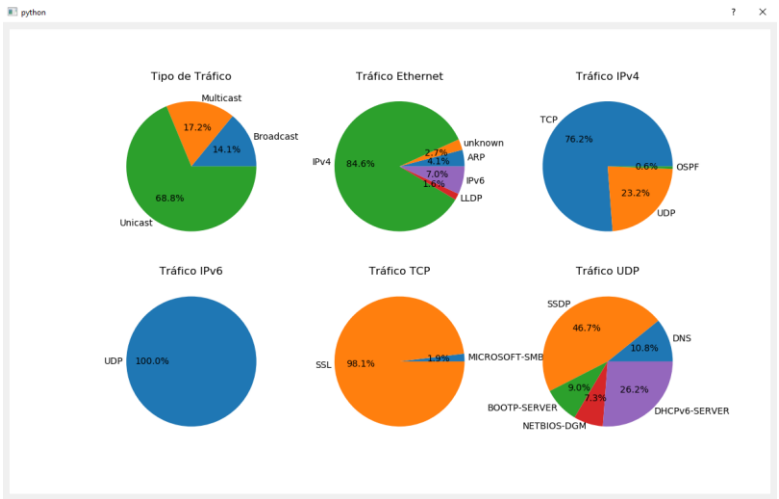


Ilustración 39. Gráficas generales

Si se quiere crear una gráfica, primero se elegirá si se quiere por número de paquetes o por Bytes, luego seleccionaremos el protocolo y por último pulsaremos el botón correspondiente de “Gráfica”. En el caso de las conversaciones previamente a pulsar la gráfica, seleccionaremos si queremos conversaciones Ethernet o IPv4.

Ejemplo:

Queremos obtener una gráfica que represente los Bytes analizados del protocolo UPD. La secuencia para seguir sería como se indica en la Ilustración 39.

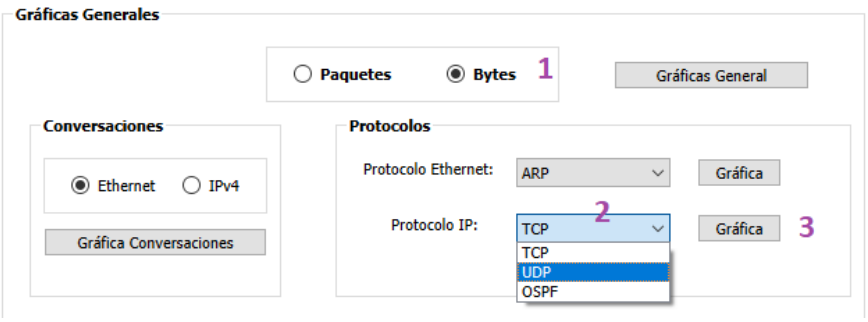


Ilustración 40. Secuencia para crear una gráfica

Cuando se pulse en “Gráfica”, saldrá una venta (Ilustración 40) con la gráfica.

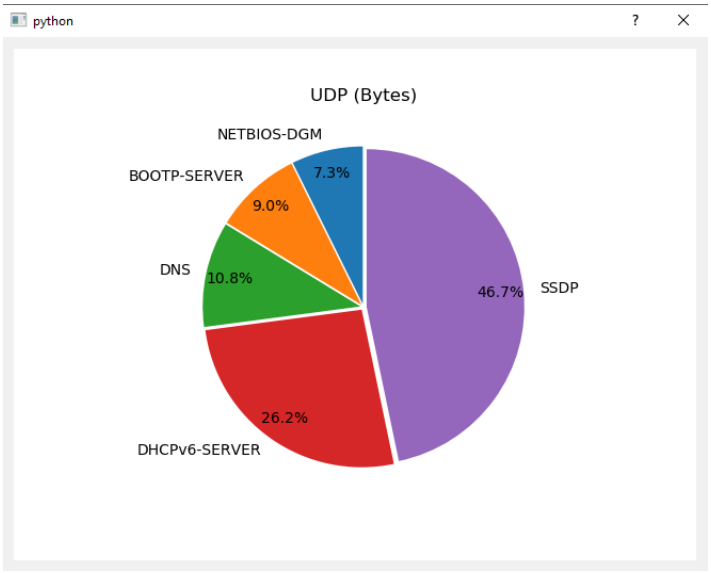


Ilustración 41. Gráfica creada por el usuario

Exportación de datos

Si el usuario quiere hacer una exportación de los datos capturados, puede hacerlo en cualquier momento en el menó “Archivo”, pulsando “Guardar datos captura” (Ilustración 41).

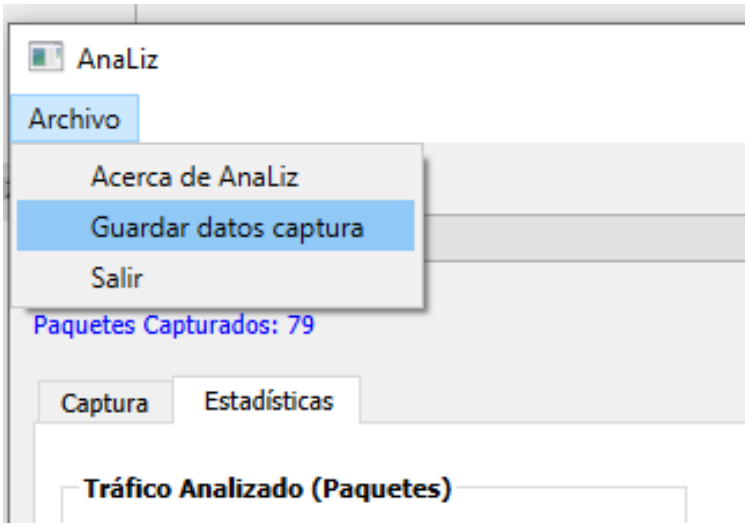


Ilustración 42. Guardar datos captura

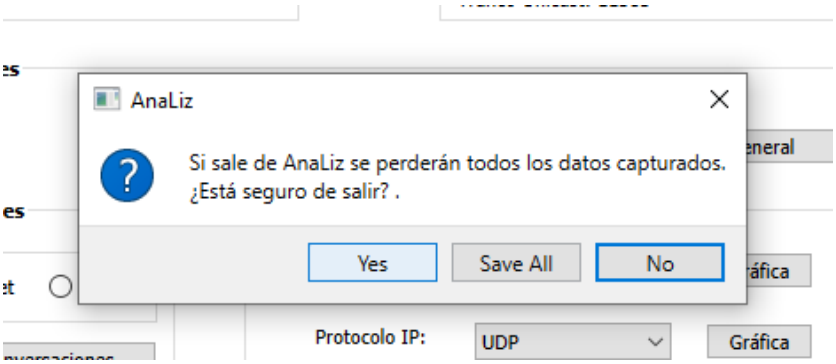


Ilustración 43. El sistema pregunta antes de deshacerse de los datos

Siempre que salgamos del programa o se comience una nueva captura, el sistema dará la opción de salvar los datos (Ilustración 42).

6. Conclusiones

Al comienzo del TFG tenía serias dudas de la elección ya que tenía la sensación de que era un proyecto muy grande para el tiempo que se estimaba. Aunque en el mundo de las comunicaciones y redes me encuentro cómodo, la parte de programación, diseño y la gestión de proyecto no tenía mucha experiencia.

A lo largo del desarrollo del TFG, y sobre todo al principio, he tenido muchos cambios de decisiones, en las que veía que el impacto era grande, como la del lenguaje de programación o la forma de representar la información.

La elección del lenguaje de programación Python considero que ha sido un acierto que, aunque tenía más experiencia en C, Python me ha permitido realizar toda la parte gráfica cómodamente que es donde tenía más dudas. Además, al ser un leguaje que está en aguje, la documentación es casi infinita.

Unas de las sensaciones que me ha dejado es que la herramienta tiene mucho margen de mejora, ya que cada vez que terminaba una fase o que me fijaba en algún aspecto de otra herramienta, quería desarrollar este trabajo.

Se dejan mejoras como análisis de velocidad, análisis de errores, perdida de paquetes, filtros o conversaciones TCP y UDP. Una mejora que veo fundamental para este tipo de herramientas es poder ver el contenido de un paquete capturado, pero la herramienta se acercaría más a un *sniffer*.

Otro aspecto que se introdujo como requisito era que la herramienta fuese multiplataforma. El objetivo está cumplido ya que la aplicación la he podido ejecutar en las plataformas Windows, Linux macOS; pero he tenido grandes dificultades en crear los paquetes de distribución, ya que el generador de Python es complicado de configurar cuando se tiene dependencias de librerías.

La parte GUI ha sido donde más tiempo se ha llevado el desarrollo. El diseño de los formularios y la programación de los diferentes *widgets* ha sido tarea arda sobre todo en la búsqueda de información, ya que el mismo formulario, pero en diferente plataforma se desconfiguraba y deformada. Se considera que una herramienta además de que funcione correctamente, también tiene que tener un aspecto limpio y atractivo para el usuario, por lo que en este sentido la aplicación es mejorable.

En el desarrollo de las gráficas también se ha empleado más tiempo del estimado, ya que la aplicación cuando tiene una gran cantidad de datos resulta difícil representarla por lo que aparece el problema del “*overlabel*”. Sin duda esta es una las principales mejoras que debe tener la herramienta.

Pero por el contrario el desarrollo en la fase de captura y clasificación del tráfico ha sido más fácil de lo que en un principio estimaba. La librería *scapy* ha facilitado bastante la tarea.

En definitiva, creo que producto resultante es una herramienta que cumple perfectamente la misión de analizar una red. El usuario ve en todo momento el tráfico, puede analizar conversaciones, se diferencian los protocolos, el tipo de tráfico y sobre todo puede realizar gráficas para un mejor estudio.

Tras la terminación del proyecto me he dado cuenta de que he adquirido experiencia sobre redes y más concretamente cómo funciona el TCP/IP. He consolidado conocimientos de comunicaciones y de programación y he probado de primera mano el impacto que tiene el análisis en un trabajo de esta índole.

Personalmente esto muy satisfecho con el resultado, y no solo por el producto resultante, sino por todos los aspectos que conlleva alrededor como la documentación, búsqueda de información, análisis, y diseño, gestión de tiempos, etc. Gracias al TFG he podido comprobar que todo lo aprendido todos estos años atrás en diferentes asignaturas, me han valido para resolver cada problema planteado y dar una solución adecuada y proporcionada.

7. Glosario

GUI: interfaz gráfica de usuario (*Graphical User Interface*).

TFG: Trabajo Fin de Grado

OSI: Modelo de interconexión de sistemas abiertos (*Open System Interconnection*)

IDE: Entorno de desarrollo integrado (*Integrated Development Environment*)

Sniffer: Analizador de protocolos.

Widget: Pequeña aplicación o programa para dar fácil acceso a funciones frecuentemente usadas.

Interface: Tarjeta de red.

Broadcast: La difusión amplia de información a una red

Multicast: Envío de la información en múltiples redes a múltiples destinos simultáneamente.

Unicast: difusión única

8. Bibliografía

INC, C. (2020). **Support - Cisco Support - Software Downloads, Product Documentation, Tools, and Cases. Cisco*, from***
https://www.cisco.com/c/en/us/support/index.html

Modelo OSI*. (2020).* Es.wikipedia.org.*, from*
https://es.wikipedia.org/wiki/Modelo_OSI

Protocolo de resolución de direcciones*. (2020).* Es.wikipedia.org.*, from*
https://es.wikipedia.org/wiki/Protocolo_de_resoluci%C3%B3n_de_direcciones

El Protocolo ARP | Redes y Seguridad*. (2020).* Redesyseguridad.es.*, from*
http://www.redesyseguridad.es/el-protocolo-arp/

Internet Assigned Numbers Authority*. (2020).* Iana.org.*, from*
https://www.iana.org/

(I), T. (2016). **Threading: programación con hilos (I). Python-para-impacientes.blogspot.com.*, from***
https://python-para-impacientes.blogspot.com/2016/12/threading-programacion-con-hilos-i.html

(2020). **Rua.ua.es.*, from***
https://rua.ua.es/dspace/bitstream/10045/11605/1/Pr2-2009-10.pdf

Protocolo de datagramas de usuario*. (2020).* Es.wikipedia.org.*, from*
https://es.wikipedia.org/wiki/Protocolo_de_datagramas_de_usuario

Primeros pasos en PyQt 5 y Qt Designer: Programas gráficos con Python*. (2019).* Medium, *from*
https://medium.com/@hektorprofe/primeros-pasos-en-pyqt-5-y-qt-designer-programas-gr%C3%A1ficos-con-python-6161fba46060

Welcome to Python.org*. (2019).* Python.org.*, from*
https://www.python.org/

***Welcome to Scapy's documentation! — Scapy 2.4.3 documentation*. (2020). *Scapy.readthedocs.io*., from**
https://scapy.readthedocs.io/en/latest/

TCP/IP y el modelo OSI | Textos Científicos*. (2006).* Textoscientificos.com*. , from*
https://www.textoscientificos.com/redes/tcp-ip/comparacion-modelo-osi

Tecnología Ethernet. - ppt descargar*. (2020).* Slideplayer.es.*, from*
https://slideplayer.es/slide/3264790/

Protocolo TCP*. (2020).* CCM*. From*
https://es.ccm.net/contents/281-protocolo-tcp

Embedding In QT5 — Matplotlib 2.1.0 documentation*. (2020).* Matplotlib.org.*, from*
https://matplotlib.org/gallery/user_interfaces/embedding_in_qt5_sgskip.html

Free Network Analyzer, Free Packet Sniffer, Capsa Free - Colasoft*. (2020).* Colasoft.com.*, from*
https://www.colasoft.com/capsa-free/

Orion Platform - Scalable IT Monitoring | SolarWinds. ***(2020).*** **Solarwinds.com,** ***from*** *https://www.solarwinds.com/es/solutions/orion*

Python. ***(2020).*** **Es.wikipedia.org.** ***from*** *https://es.wikipedia.org/wiki/Python*

Common Ports**. (2020).** ***Web.mit.edu*****. from** https://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-sg-en-4/ch-ports.html

AJAX Loading Images - Free Online GIF Icons Generator. ***(2020).*** **Ajaxloadingimages.net.,** ***from*** *http://ajaxloadingimages.net/*

14.04), H., & Compiler, T. (2016). **How to install PyQt5 in Python 3 (Ubuntu 14.04). Stack Overflow.** ***from*** *https://stackoverflow.com/questions/36757752/how-to-install-pyqt5-in-python-3-ubuntu-14-04*