Aplicació gràfica per analitzar el tràfic de xarxa (Sniffer), amb entorn GNU

Nadal Salamanca Nadal
ETIS
Consultora: María Isabel March i Hermo

13 de Gener de 2013

RESUM DEL TREBALL

El Treball Final de Carrera (TFC), presentat a continuació, es troba dintre l'àrea de Xarxes i Computadors, i el seu objectiu es realitzar una aplicació de tipus gràfic, amb entorn GNU(GNU is Not Unix), que sigui capaç d'analitzar tot el tràfic que es produeix dintre de una xarxa. Aquest tipus d'aplicacions es coneixen com Sniffers(Detectors).

L'eina que es pretén desenvolupar, està orientada a usuaris amb coneixements basics de xarxes, de forma que l'aplicació ha de esser tot l'interactiva que es pugui, per tant de permetre que qualsevol usuari la pugui utilitzar, tant per fins educatius (veure quin es el contingut dels paquets que viatgen per la xarxa) com a nivell professional, per ajudar a la detecció d'intrusions a la xarxa, i monitorin-te el tràfic de la xarxa per a detectar anomalies.

Aquesta aplicació es desenvoluparà en llenguatge JAVA, per tant es podrà executar amb qualsevol maquina que disposi de una maquina virtual Java, independentment del sistema operatiu utilitzat (Windows, OS Apple o diferents versions de Linux o Unix).

Serà capaç de capturar qualsevol tipus de paquet que viatgi per la xarxa, emperò nomes analitzarà els següents protocols:

- Trames Ethernet.
- Paquets IP, ICMP, ARP,SNMP.
- Segments TCP i UDP.

També oferirà la possibilitat de guardar la captura en un fitxer, que es podrà recuperar de forma diferida per analitzar-lo, i a més fer estadístiques de forma gràfica.

INDEX DE CONTINGUTS.

INDEX DE FIGURES

1 - INTRODUCCIÓ.

Un Sniffer, es un eina que llegeix el tràfic que passa per la tarja o targes de xarxa que tenim instal·lades al nostre ordinador. Permet identificar de quin tipus es aquest tràfic, i filtrar el que ens interessa.

El tràfic de una xarxa de tipus Ethernet, esta format per trames, aquestes trames contenen diferents tipus de paquets, cada un amb un format diferent, el Sniffer també ens permet editar cada una de aquestes trames per a veure el seu contingut.

La utilitat d'aquesta eina es a mes del esmentat, poder detectar colls de ampolla, o detectar intrusions a la nostra xarxa, de fet es utilitzat tant per finalitats educatives com per a administradors de xarxa per la seva feina diari.

L'eina desenvolupada es centrarà amb l'anàlisi de les trames que s'utilitzen en el model TCP/IP. .

1.1 Justificació del TFC i context en el qual es desenvolupa: punt de partida i aportació del TFC.

L'objectiu d'aquest treball, es assimilar les diferents assignatures que hem fet durant la carrera, i aplicar-les a un treball. Jo he triat el camp de les xarxes, perquè es el que mes m'agrada de l'informàtica, i crec que es el que te un desenvolupament mes gros. Tot i que la meva feina habitual no hi te res a veure. Som cap de projectes de desenvolupament de fa 20 anys, però les meves inquietuds sempre han estat de la part de sistemes.

1.2 Objectius del projecte:

Els objectius d'aquest projecte es desenvolupar un eina que permeti analitzar quin es el tràfic que es produeix dintre una xarxa especifica. Es que es pugui representar de forma gràfica es un afegit que o fa mes atractiu per l'usuari. De cap manera es pretén analitzar el rendiment de la xarxa, sinó veure que es que passa per dintre, es a dir quins paquet circulen, i de quins tipus, si son IP, ICMP o ARP, i dintre aquets si porten encapsulats segments TCP, UPD o SNMP.

Una vegada identificat el paquet, el que fa es interpretar tots el camps que porta la seva estructura, per a poder expandir-los de forma que el usuari vegi el seu contingut, de una forma clara. L'aplicació ha de esser capaç de identificar la informació continguda a cada paquet, i mostrar-la de forma estructurada, de manera que el usuari pugui identificar fàcilment el contingut de cada un dels camps que formen un paquet que pertany a un dels protocols analitzats.

En principi, a grans trets, s'hauria d'identificar quins son el interfícies de xarxa que disposa el dispositiu amb el que fèiem servir l'aplicació, i a partir d'aquí triar quins son el tipus de tràfic a analitzar. També ha d'incorporar funcionalitats, per treure resums del tràfic analitzat, tant de forma textual com amb forma gràfica (tipus de paquets capturats). De forma gràfica ha de mostrar els paquets capturats per cada tipus de protocol en forma de barres 3D.

Com s'ha dit a l' introducció l'abast d'aquest projecte es detectar trames Ethernet, i dintre aquestes els següents tipus de paquets:

- Paquets IP, ICMP y ARP
    - Segments TCP UDP i SNMP.

1.3 Enfocament i mètode seguit:

En primer lloc, el que vaig fer es veure quines son les eines mes utilitzades, que tinguin funcionats similars al projecte que volem assolir
Hi han infinitat d'eines similars dintre el mercat, el que he fet, es dintre el ampli ventall d'oferta, es escollir-ne tres per ordre de dificultat, de la mes complerta fins a la mes senzilla, per tal de veure fins on podia arribar amb els coneixements adquirits i amb el temps disponible.

S'han instal·lat i provat les eines:

-WireShark
-COMMVIEW
-PTraf

El seu funcionament esta descrit al capítol 2.

He hagut de repassar tots el materials de la carrera, que tenen que veure amb les tasques del projecte, Programació orientada a objectes, Enginyeria del programari, Xarxes. He ampliar informació cercant a Internet, sobre els protocols de xarxa, exemples, rutines i llibreries Java que em pogués facilitar la feina

1.3 Metodologia de desenvolupament.

Per elaborar el pla de treball es segueix la metodologia inspirada en el Rational Unified Process , que es composa de les següents fases:

1. **Recollida de requisits**: especificació de funcionalitats i tipus d'usuaris els que va dirigida l'aplicació. Per aquest apartat s'han estudiat diferents eines existents per a veure que tenen en comú, i quin tipus d'informació mostren, així com el grau de dificultat que te el seu us.

2. **Anàlisi i disseny**: creació de les classes i diagrames que representaran l'esquelet de l'aplicació, amb el llenguatge UML. A partir del casos d'us recollits a la fase 1, dibuixar els corresponents diagrames i veure les seves interrelacions. També identificar quines son les classes necessàries i el seu contingut.

3. **Realització**: implementació de l'aplicació. Aquí es on es van construint els diferents mòduls en els que s'ha dividit l'aplicació. Primer l'interface d'usuari, i a partir de aquí com anar omplint cada una de les funcionalitats, un mòdul per triar l'interface, un altre per el filtratge, un mòdul per capturar els paquets, un per analitzar el seu contingut, un altre per mostrar la informació, i finalment un mòdul per mostrar les gràfiques.

1.4 Planificació del projecte.

Per establir un calendari de realització del projecte, s'ha dividit aquest amb tasques, amb un desglossament d'aquestes i una previsió temps, per tant d'ajustar-lo al disponible.

Tasca 1:

Temporització: 2 setmanes (1 d'octubre a 14 d'octubre).
Descripció: Recollida d'informació que pugui ser rellevant per dur a terme el projecte:

1. Tenir una visió general sobre la funcionalitat d l'eina d'anàlisi.
2. Recopilar informació de les eines mes usuals, existents al mercat, que facin les mateixes funcionalitats que volem obtenir.
3. Obtenir informació sobre les diferents tipus de trames de xarxa que haurem d'analitzar.

Tasca 2:

Temporització: 2 setmanes (15 d'octubre al 28 d'octubre).
Descripció: anàlisi i disseny

1. Classes més importants de l'aplicació en el model del domini i el model de negoci.
2. Definició els casos d'ús.
3. Requisits de la interfície –funcionals i no funcionals- i els requisits de funcionament, amb la descripció textual dels casos d'ús.

Tasca 3.

Temporització: 1 setmana (29 d'octubre al 4 de Novembre).
Descripció: recerca de recursos externs.

- Llibreries que puguin esser d'utilitat.
- Preparar framework de Java.
- Documentació i estudi de les llibreries.

Tasca 4.

Temporització: 6 setmanes (5 de Novembre al 23 de desembre).
Descripció: primera implantació.

- Disseny d'interfície d'usuari.
- Programació de les classes.
    - Identifica l' interfase de xarxa
    - Preguntar quin interfase volem estudiar
    - Preguntar quint tipus de paquet
    - Captura de un paquet.
    - Anàlisi del paquet.
    - Mostrar el paquet de forma intel·ligible.
    - Guardar el log.
    - Mostrar el resum de forma gràfica

- Integració amb les llibreries
- Programar les interfícies d'usuari.

Tasca 5.

Temporització: 1 setmana (24 de desembre al 30 de desembre).
Descripció: primera versió funcional.

- Encaixar tots els mòduls.
- Preparar eina d'instal·lació.
- Proves d'integració.

Tasca 6:

Temporització: 2 setmanes (31 de desembre al 13 de gener).
Descripció: Preparar la presentació.

- Retocs estètics aplicació
- Període proves
- Memòria.

Desviació temporal al pla de treball inicial

Aquest pla de treball inicial ha sofert una desviació, en la tasca 4, que ha fet impossible implantar la funcionalitat de guardar el log a un altre fitxer, ja que han aparegut molts de problemes amb la implementació del thread. Aquests problemes han estat deguts a falta de experiència amb programació en Java, ja que en principi totes les funcionalitats de captura estaven incloses en la classe principal, una vegada feta l' integració dels mòduls (Tasca 5), s'ha vist que era pràcticament impossible controlar la finalització de captures, i he decidit implementar el thread. Aquest canvi d'estratègia ha portar molta mes feina de la prevista, i ha fet que la funcionalitat prevista de guardar el log a un fitxer triat per l'usuari, hagi estat impossible d'acabar.
Per tant vist, que era mes important que es captura sen be els paquets, i que els mostres de forma correcte, el que podia deixar de fer era lo del log. De fet canviant el nom al log per defecte, es possible simular aquesta funcionalitat de forma manual.
També s'ha hagut de afegir una nova funcionalitat no prevista, que es la de reiniciar la sessió, ja que essent una eina orientada als events, i no procedural, no hem ha quedat mes remei que preveure una posta a 0 de la sessió per a poder continuar fent captures.
Això ha implicat ampliar la tasca 5 amb 1 setmana i acurtar la tasca 6 amb 1 setmana dedicant-li moltes mes hores de les previstes.

1.5 Productes obtinguts

• Un analitzador gràfic de xarxa amb entorn GNU, capaç de capturar el tràfic que es produeix a la xarxa a la que estaiem connectats, i amb un filtratge establert per l'usuari, identificar els paquets que hi viatgen i expandir els camps de cada protocol.

• La memòria del TFC (aquest document).

• Manual de instal·lació y usuari (inclòs en la memòria).

1.6 Breu descripció dels altres capítols de la memòria.

Els següents capítols expliquen cada una de les etapes de l'aplicació desenvolupada:

2 Conceptes teòrics previs.
Repàs de teoria i recerca d'informació.
3 Anàlisi i disseny del projecte.
Recollida de requisits i creació de diagrames UML.
4 Implementació del projecte.
Forma en que s'ha implementat el projecte.
5 Proves.
Realització de proves per veure que el programa fa el que volem.
6 Manual d'instal·lació i d'usuari.
Confecció de un manual de instal·lació i d'us.
7 Conclusions.
Conclusions obtingudes a la finalització del projecte i millores possibles d'aquest.
8 Glossari.
9 Bibliografia.
Fonts d'informació utilitzades.
10 Annexes
Descripció dels protocols analitzats.

2 - CONCEPTES TEÒRICS PREVIS.

Per al desenvolupament i l'anàlisi del projecte s'ha hagut de repassar i ampliar conceptes teòrics, que fan referència al model TCP/IP i a les eines existents en el mercat que tenen funcionalitats semblants al que volem obtenir. A mes cal repassar tot el material de les assignatures que fan referència a programació orientada a objectes, i les de enginyeria del programari.

El model TCP/IP.

El Departament de Defensa dels EEUU va crear el model TCP / IP perquè necessitava una arquitectura que pogués connectar múltiples xarxes i que tingués la capacitat de mantenir connexions tot i que una part de la subxarxa estigués danyada per motiu d'alguna guerra nuclear. Això va portar a la creació del projecte ARPANET (promogut i finançat pel DARPA, secció del Departament de Defensa dedicada a la investigació). D'aquest projecte va sorgir el model de comunicació entre ordinadors de diferents xarxes basat en l'intercanvi de paquets.

Diverses universitats americanes van modificar aquest model de comunicació,creant un sistema propi, anomenat Internetting que, en anar ampliant a xarxes cada vegada més grans, va passar a anomenar-se Internet. La base d'aquesta Xarxa de xarxes va ser el model TCP / IP, el qual està basat en el tipus de xarxa de commutació de paquets i té quatre capes: la capa d'aplicació, la capa de transport, la capa d'Internet i la capa de xarxa.

El model TCP / IP tenia dos objectius:

1) Construir una interconnexió de xarxes que proporcionés serveis de comunicació universals: una xarxa, o internet.

2) Inter connectar diferents xarxes físiques per formar el que l'usuari lis embla una única i gran xarxa.

Tal conjunt de xarxes Inter connectades es denomina "Internetwork" o internet. Ethernet és la tecnologia més estesa a tot el món per a la implementació de xarxes d'àrea local. Gestiona l'intercanvi de dades entre ordinadors, podent usar diferents protocols com TCP / IP, Netware, AppleTalk, VINES, etc. El més estès és la pila de protocols TCP / IP (Transport Control Protocol / Internet Protocol), que és un model pràctic, a nivell mundial, i un suport per a la intercomunicació de tot tipus de xarxes i sobre el qual s'ha desenvolupat Internet. El model de referència TCP / IP i la pila de protocol TCP / IP fan que sigui possible la comunicació entre dos ordinadors en qualsevol lloc del món. La pila TCP / IP es diu així per dos dels seus protocols més importants: TCP ("Transmissió Control Protocol") IP ("Internet Protocol"). TCP / IP, com la majoria del programari de xarxa, està modelat en capes. Aquesta representació condueix al final pila de protocols. Els protocols d'Internet es modelen en quatre capes

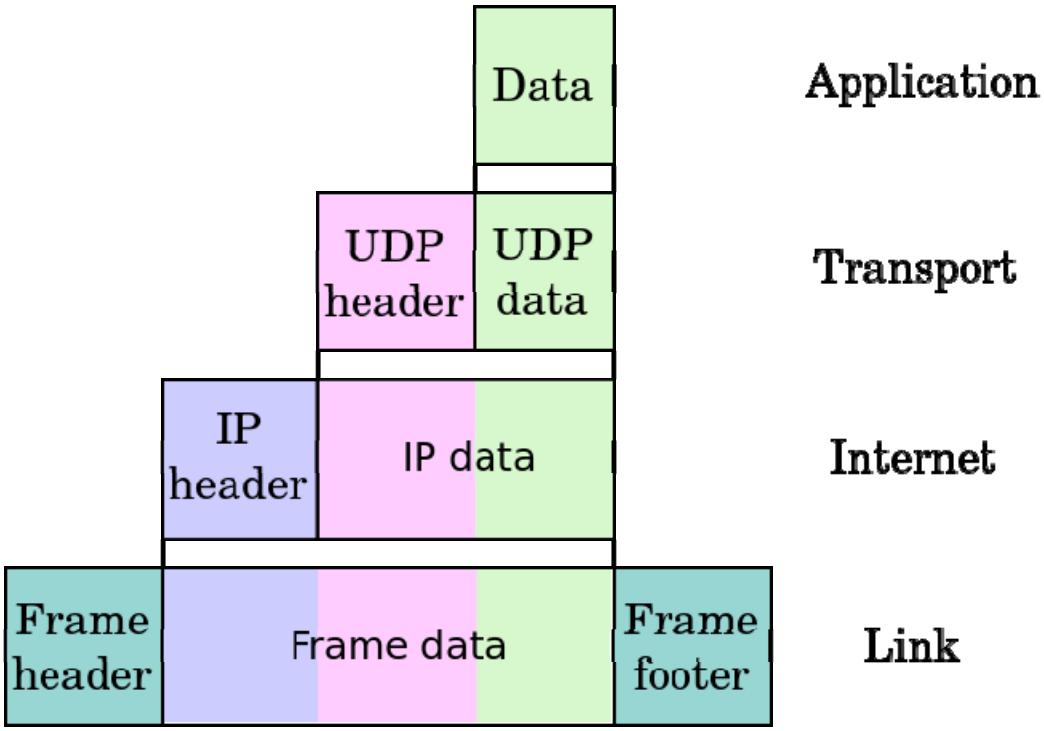

Figura 1 Pila de protocols

Cada una d'aquestes capes porta encapsulada l' informació de les següents, es a dir, el que viatge per la xarxa son trames Ethernet, que correspon a la capa 1 (Network Interface and Hardware), dintre aquestes trames hi trobem paquets de tipus IP, ICMP o ARP, que pertanyen a la capa de Internetwork, a la vegada dintre aquests paquets podem trobar dades de tipus TCP o UDP, que pertanyen a la capa de transport, i per últim dintre aquets paquets TCP o UDP, hi viatgen els paquets de la capa d'aplicació, que poden ser de tipus HTTP, FTP, SNMP.
La informació continguda a cada un dels protocols que conformen les diferents capes, està descrita al capítol 9 (**Annexes**), d'aquesta memòria.

Per exemple en el camp protocol de la Capçalera IP d'un Datagrama IP es codifica el valor del protocol de nivell superior que viatjarà en el seu camp de dades segons el IANA, organisme encarregat de l'assignació de nombres als protocols, podem destacar els següents valors (en decimal) per aquests protocols de nivell superior. Aquets valors estan definits a la taula que es pot trobar a la següent URL.

http://es.wikipedia.org/wiki/N%C3%BAmeros_de_protocolo_IP

Eines estudiades.
A més s'ha recollit informació d'altres aplicacions que fan el mateix que volem fer amb el nostre projecte, com son WireShark ,(que es la que varem utilitzar a l'assignatura de xarxes), COMMVIEW i PTraf , he decidit aquestes tres per ordre de prestacions, de mes complerta a mes senzilla.
El criteri utilitzat es dintre del gran ventall d'eines existents ha estat fer una tria per ordre de completesa , de mes complerta a mes simple:

- WHIRESHARK (abans Ethereal)) http://www.wireshark.org/)
- COMMVIEW. http://www.tamos.com/products/commview/
- PTRAF. ftp://iptraf.seul.org/pub/iptraf/

WIRESHARK
Primer se ha instal·lat i estudiat el WireShark, que es l'eina que varem utilitzar a la assignatura de xarxes, per a recordar el que feia i com ho feia.
El WireShark, es l'eina mes extensa per a anàlisi de xarxes, i de fet es el mes complet dels 3 que he vist.
Com a característiques principals, te que es multi plataforma, es capaç d'analitzar centenars de protocols, te una gran quantitat de filtres per a aplicar, i a mes es capaç d'actuar damunt diferents tipus de xarxes, com a (Ethernet, IEEE 802.11, PPP/HDLC, ATM, Bluetooth, USB, Token Ring, Frame Relay, FDDI) i altres depenent de la plataforma on esta instal·lat.
També es capaç de desencriptar bastants de protocols, entre els quals es troben IPsec, ISAKMP, Kerberos, SNMPv3, SSL/TLS, WEP, i WPA/WPA2

Breu descripció del funcionament:

El WireShark, s'inicia amb aquesta pantalla on ens mostra els interfases de xarxa que disposa l'equip que utilitzem, així com les opcions de captura que ens interessen.

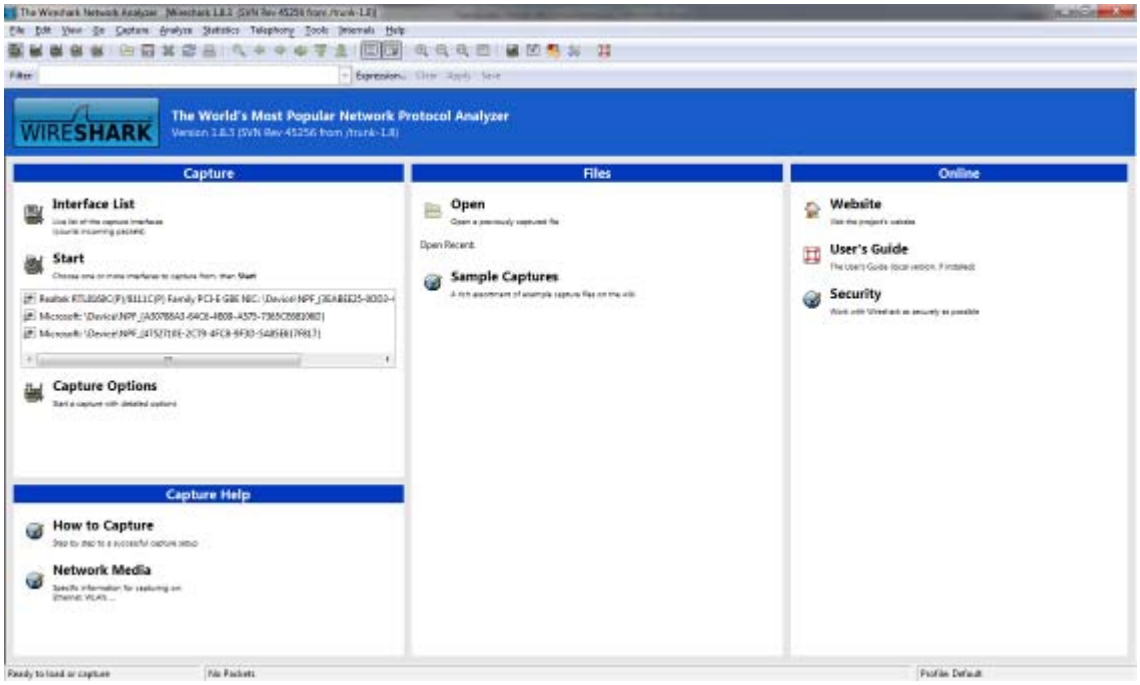

Figura 2 Inici de WhireShark

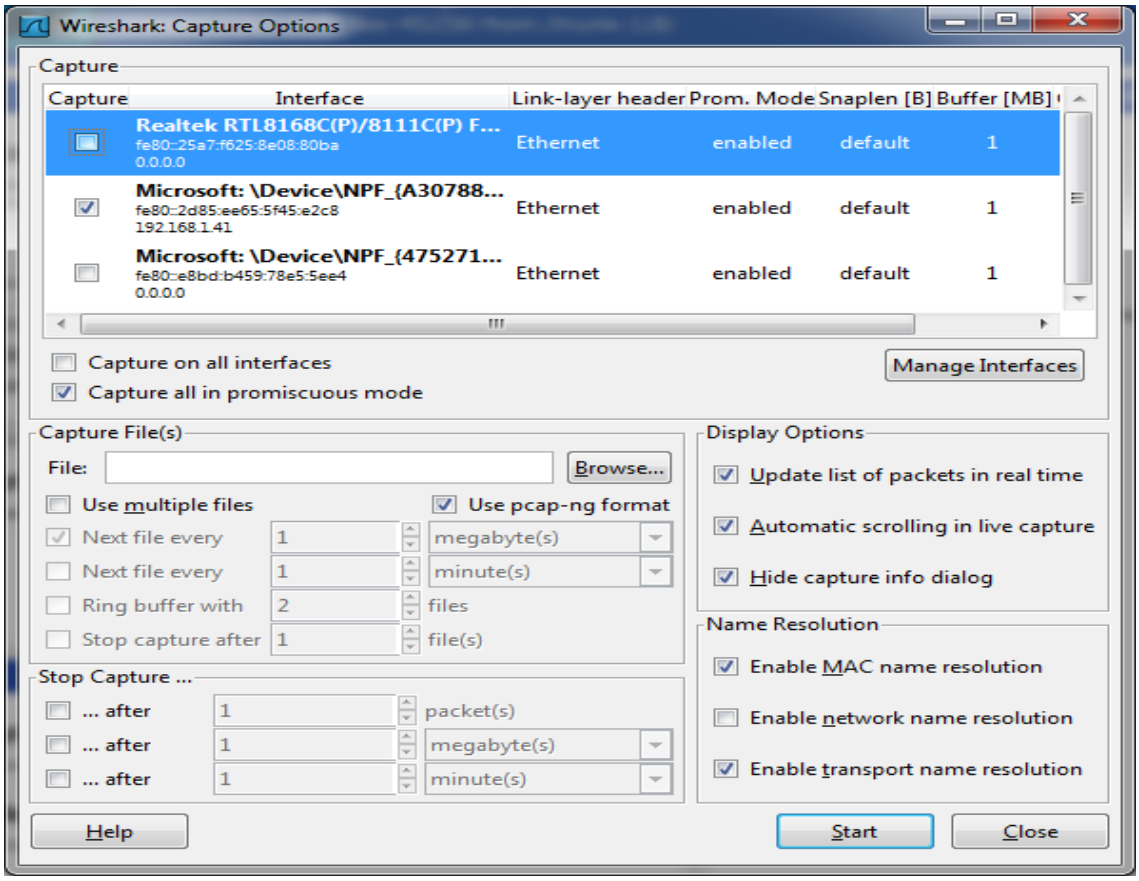

Figura 3 Opcions de WhireShark

Una vegada triats el interfases que volem analitzar, donant a l'opció start, comença la captura.

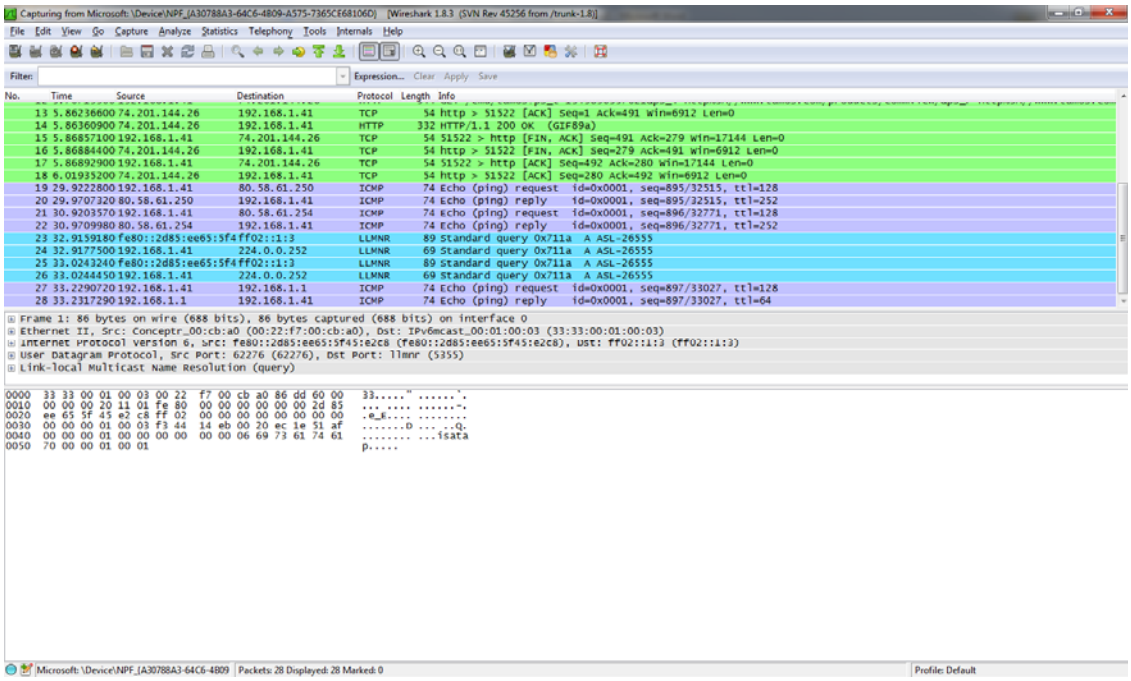

Figura 4 Captura de WhireShark

I des de aquí podem accedir a les opcions de filtratge

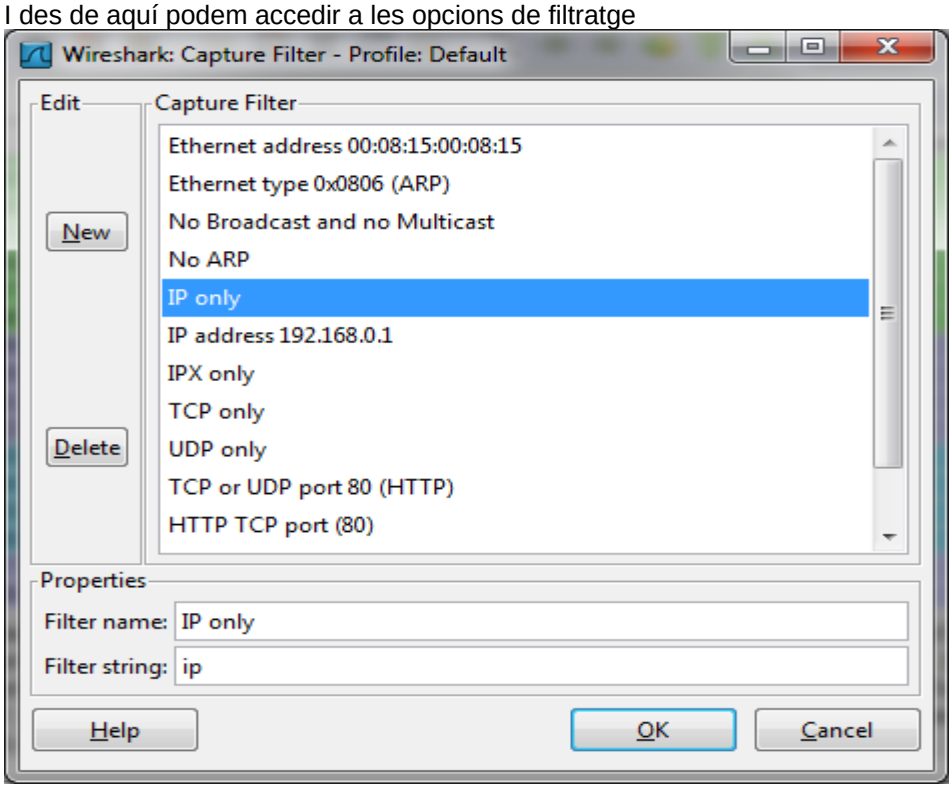

Figura 5 Filtratge de WhireShark

Quant cliquem damunt un paquet, podem veure el seu contingut.

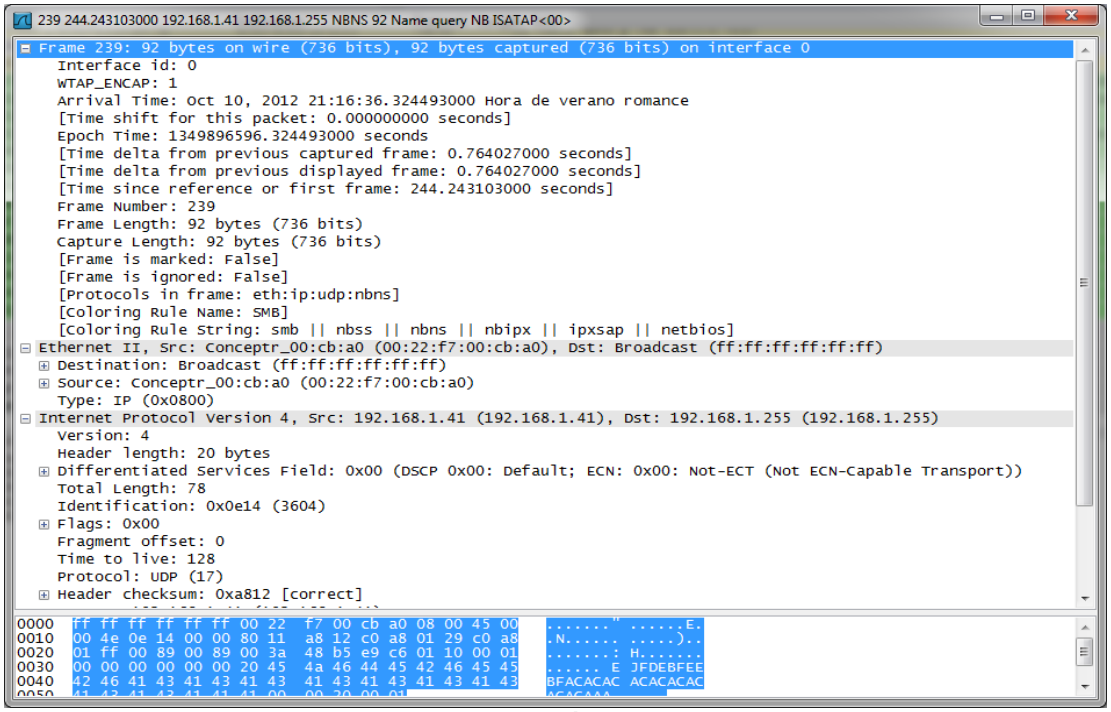

Figura 6 Expansió de paquet de WhireShark

d'una manera molt detallada i intuïtiva.
Es tracta de un eina molt complerta, que també porta diferents informacions de forma gràfica.

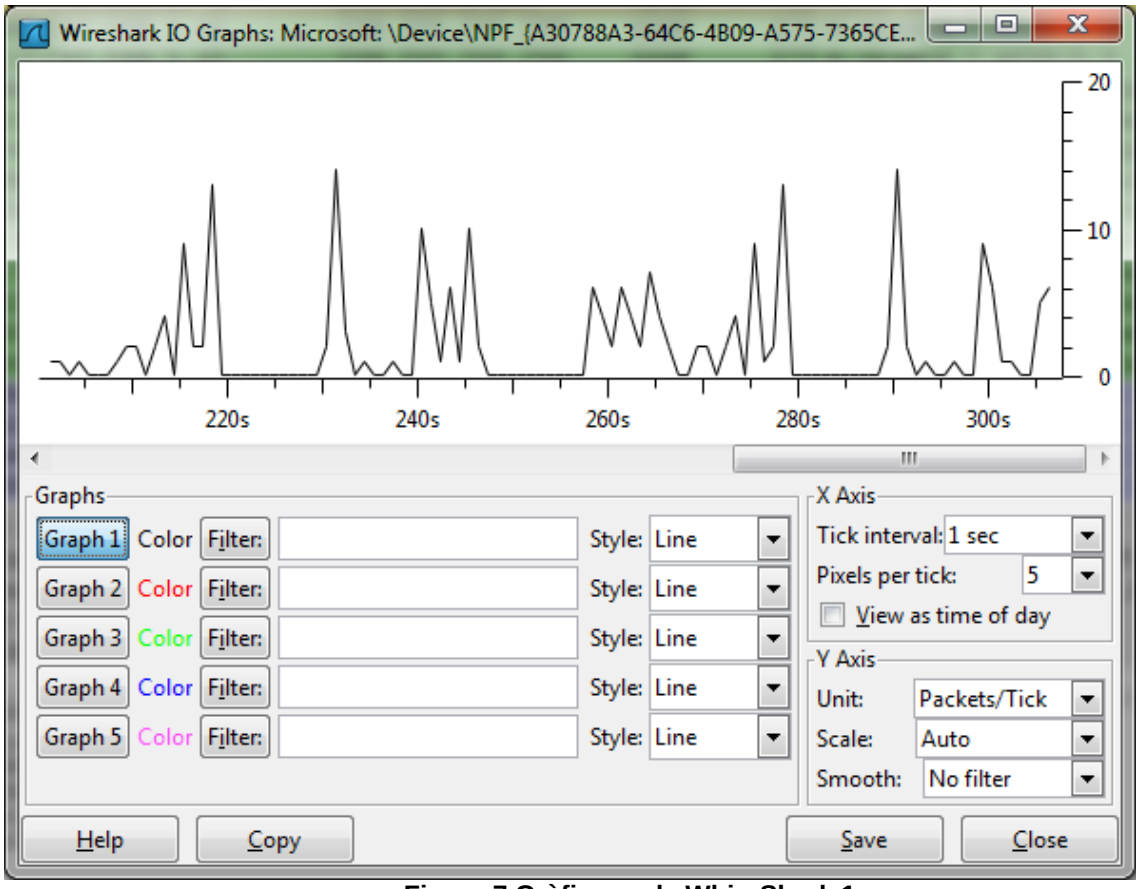

Figura 7 Gràfiques de WhireShark 1

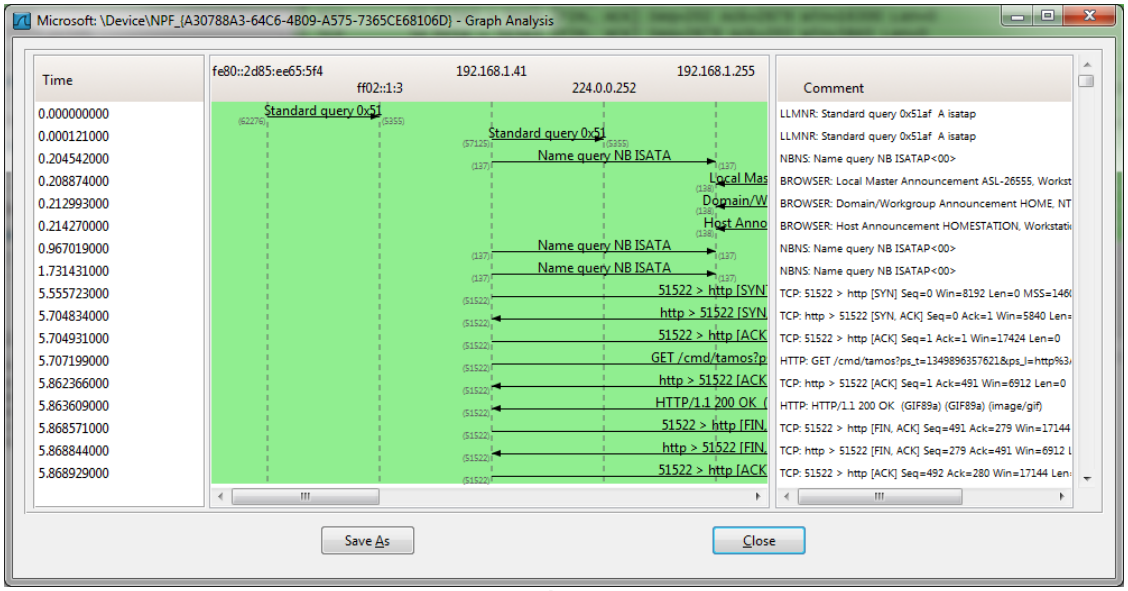

Figura 8 Gràfiques de WhireShark 2

La seva llista de protocols suportats es immensa

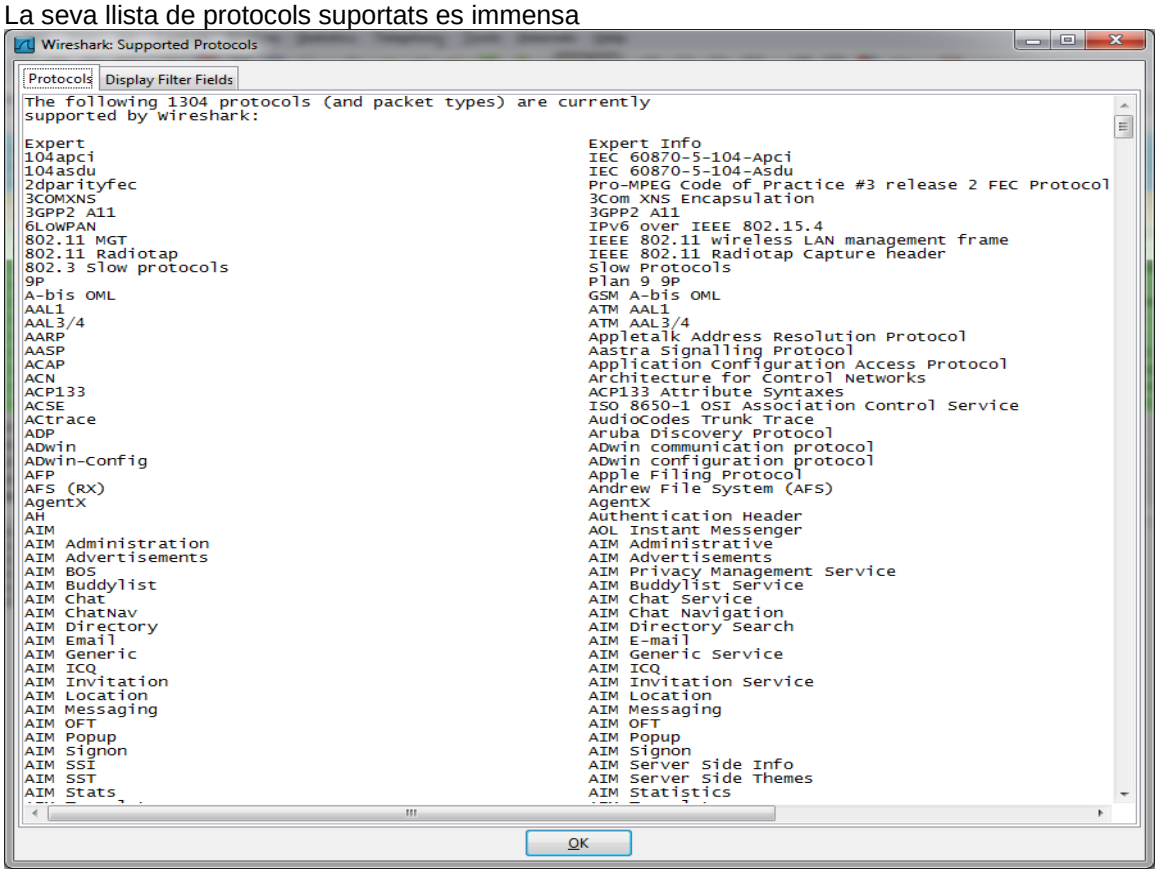

Figura 9 Llista de protocols de WhireShark

L'altre eina estudiada a estat COMMVIEW.

Aquesta eina a part de tenir casi les mateixes funcionalitats que WireShark, te un aspecte que m'ha agradat molt, perquè s'atraca al que jo pretenc amb el meu projecte, i es la part d'estadístiques en forma gràfica, on supera clarament a WireShark

Si be el seu disseny es mes senzill, y les opcions mes reduïdes, jo diria que esta dirigida a tots tipus d'usuaris. Tot i que el seu ventall de protocols que analitza també es molt ampli.

Aquí una captura de pantalla amb els protocols que suporta.

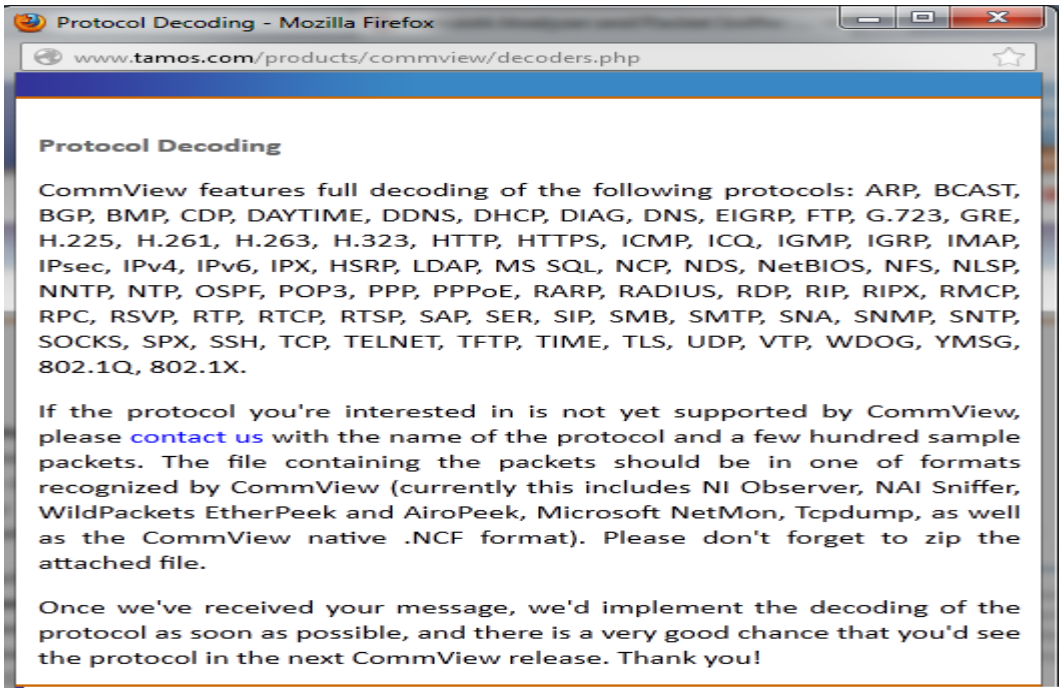

Figura 10 Llista de protocols de Commview

El seu funcionament es molt intuïtiu, a continuació he afegit unes captures de les pantalles mes importants. Aquesta es la pantalla principal de l'aplicació.

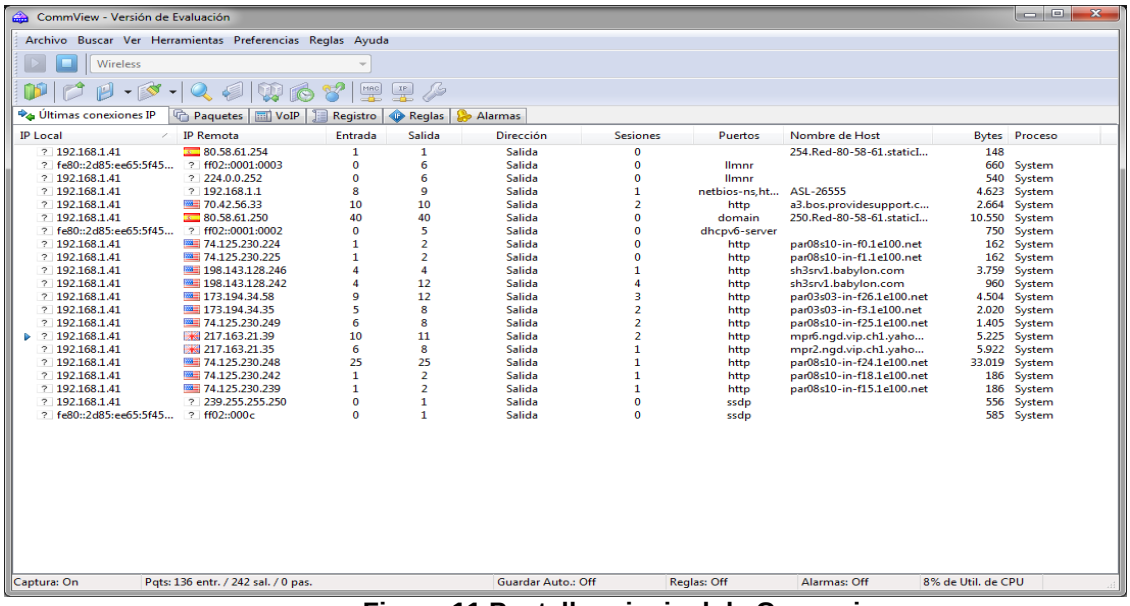

Figura 11 Pantalla principal de Commview

I tot seguit algunes de les sortides gràfiques a les que em referia.

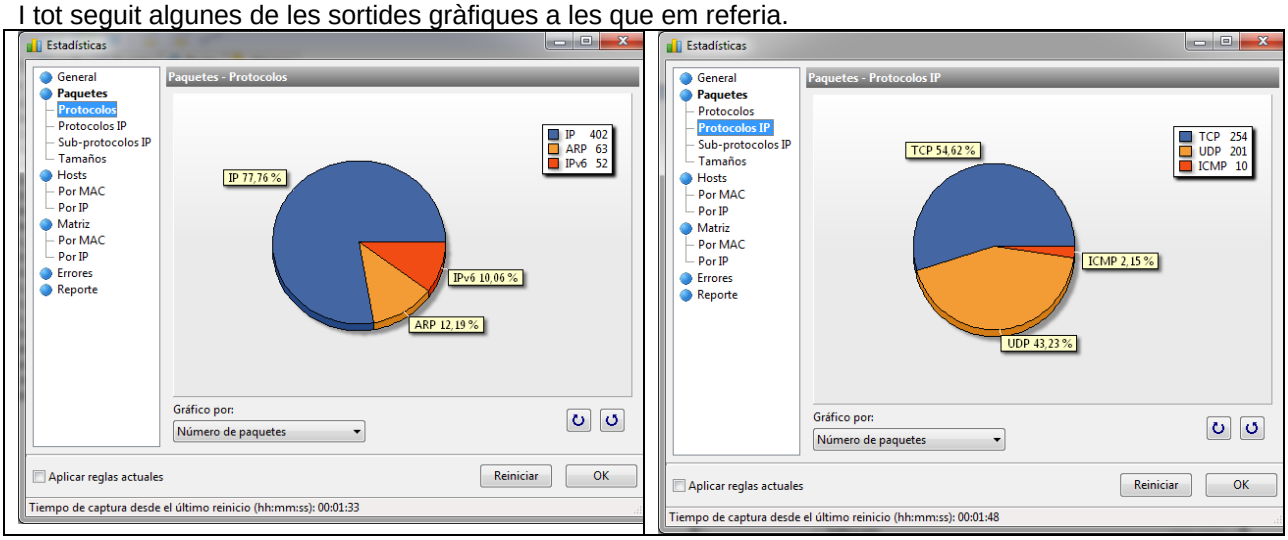

Figura 12 Gràfiques de Commview

Ptraf.

La darrera eina estudiada es diu PTraf
Aquesta es la mes senzilla de totes, i també l'única de software lliure, i a més nomes es pot instal·lar sota Linux.
Les seves característiques son mes reduïdes, ja que els protocols suportats son els següents: IP,TCP ,UDP ,ICMP ,IGMP ,IGP ,IGRP ,OSPF ,ARP ,RARP, incorpora un mòdul d'estadístiques, i els tipus de xarxa que analitza son els següents:

- loopback local
- Ethernet
- FDDI
- SLIP
- PPP asíncrona
- PPP síncron sota ISDN
- ISDN amb encapsulació IP Raw
- ISDN amb encapsulació HDLC Cisco
- IP en Línia Paral·lela

Pantalla de benvinguda

Figura 13 Pantalla d'inici de Ptraf

Sistema de menús per a triar el interfase que volem monitoritzar.

Figura 14 Sistema de menús de Ptraf

Pantalla de resultats

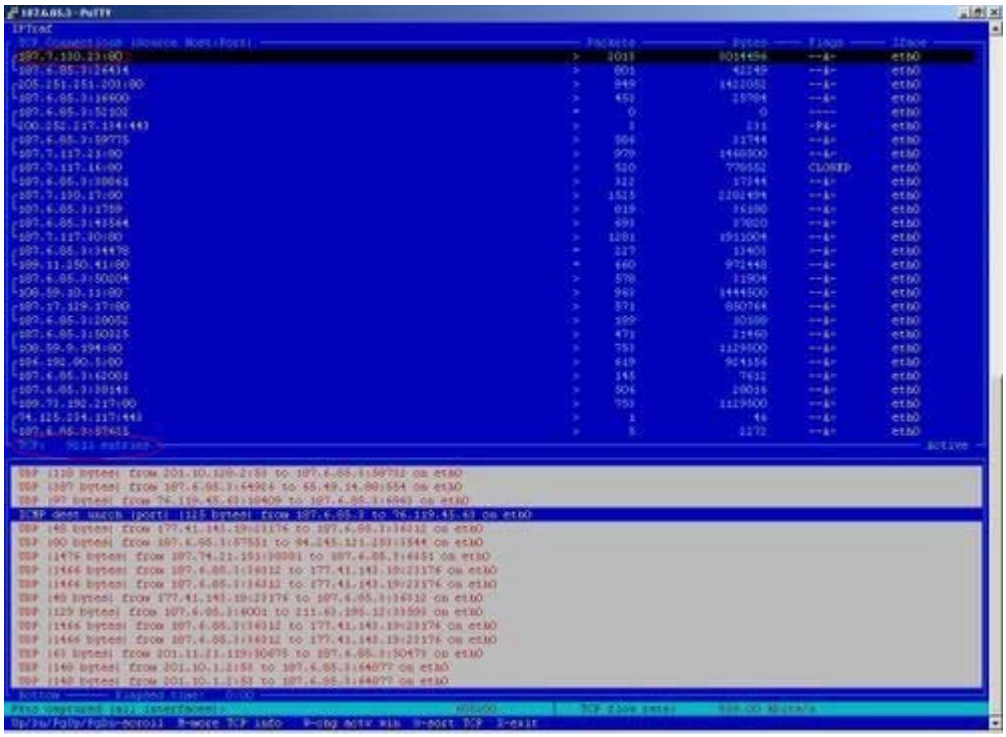

Figura 15 Pantalla de captura de Ptraf

Una vegada vistes aquestes eines, podem fer una aproximació al que volem. De fet s'ha triat l'eina mes completa (WireShark), una mitjana (CONVIEW) i la mes senzilla Ptraf.
La seqüència sempre es la mateixa:

- 1 Triar el interfase
- 2 Opcions de filtratge
- 3 Arranca el rastreig.
- 4 Veure un paquet
- 5 Veure estadístiques.

2. ANALISI I DISENNY DEL PROJECTE.

L'anàlisi sempre es la primera feina en el desenvolupament de qualsevol projecte. Consisteix en traduir el requisits que s'ha recollit al voltant de les premisses inicials i mitjançant diverses tècniques, a un llenguatge i estructura apropiada per iniciar el projecte.
Una forma de fer-ho es utilitzar els models i convencions UML, per definir las classes fonamentals i les interrelacions entre aquestes. Després es detallaran els casos d'us que seran la base per el desenvolupament del projecte.

2.3 Model de domini

De la recollida i documentació de requisits s'ha extret el següent model de domini que recull les classes més importants:

El model de domini proposat es el següent:

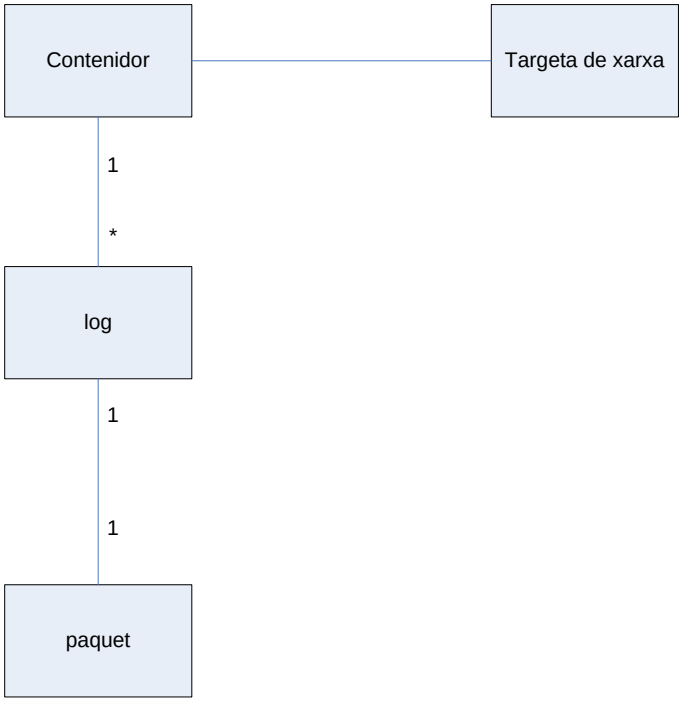

Figura 16. Model de domini

La classe "paquet" encapsula els diferents nivells de la xarxa, a cada nivell es troben els diferents protocols.
La classe "targeta de xarxa" es l'origen de les dades de captura.
La classe contenidor es on s'emmagatzemen les dades de cada una de les captures, es una estructura adaptada per fer el tractament del paquets (conte els camps comuns a cada paquet, i les dades del paquet).
La classe log, es on s'emmagatzemen els paquets capturats, ja sigui en el disc dur o en memòria.

2.4 Diagrama Estàtic de classes

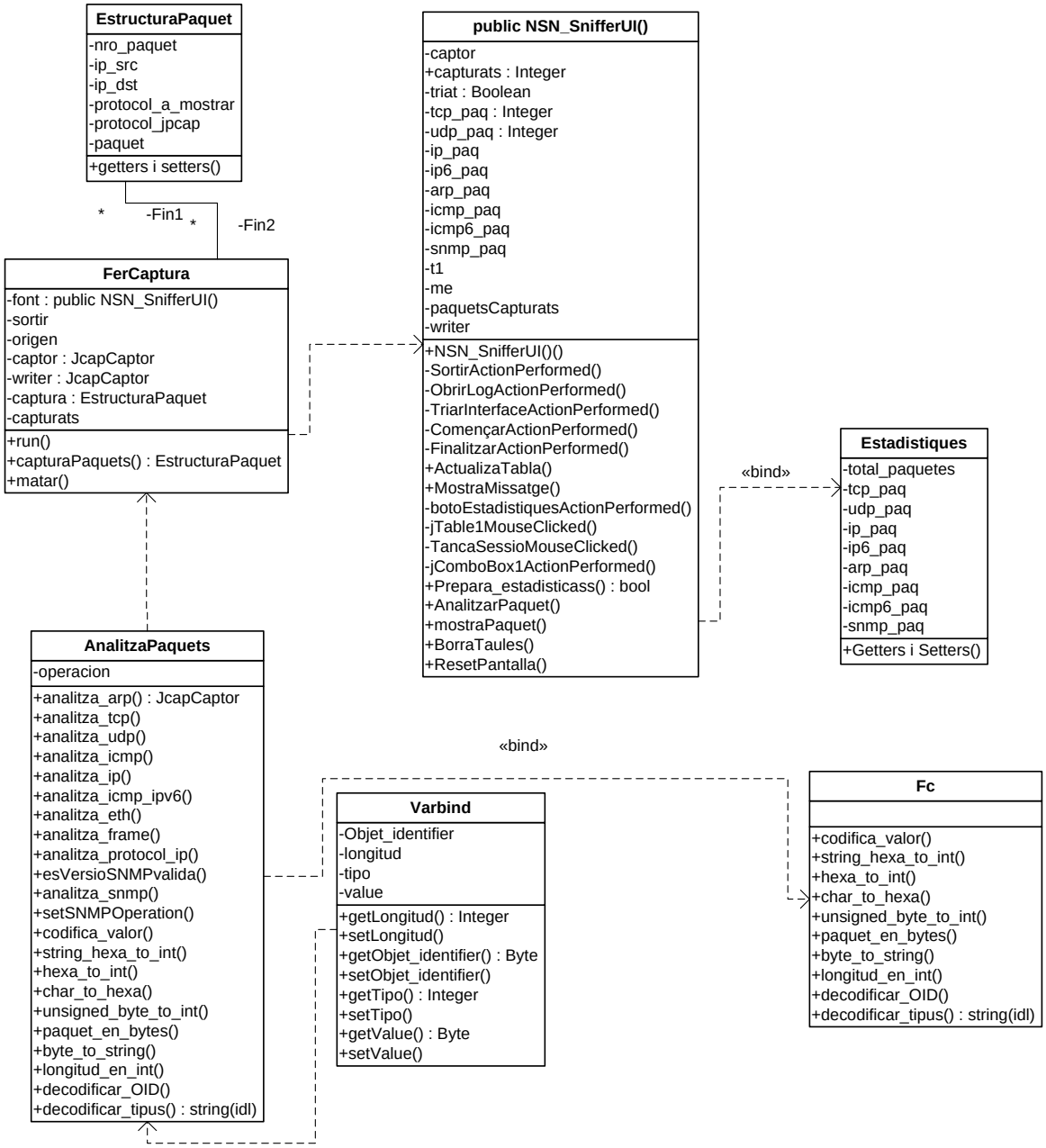

Figura 17 Diagrama de classes

La classe principal **NSN_SnifferUI**, es la que governa la execució del programa i que crida a la classe **FerCaptura** com a Thread.
La classe **FerCaptura** utilitza com a contenidor la classe **EstruturaPaquet**, i a mes utilitza la classe **AnalitzaPaquets**, per analitzar cada un dels paquets capturats, per a preparar la seva visualització.
La classe **AnalitzaPaquets**, conté tots el mètodes necessaris per analitzar cada un dels protocols i s'ajuda de la classe **Fc** (funcions comuns), per formatejar la sortida, així com la classe **Varbind** per estructurar tipus de variable complexos.
La classe **Estadístiques** es el contenidor de les variables per a treure les gràfiques.

2.5 Diagrama de casos d’us:

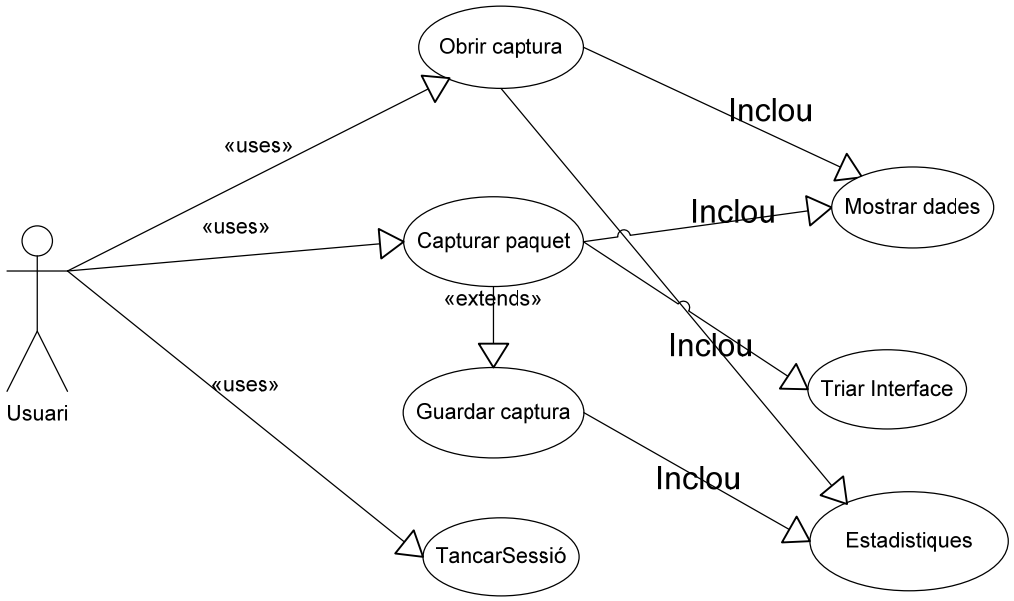

Figura 18 Casos d'us

El únic actor es l'usuari. Aquest pot ordenar els diferents usos, com Obrir captura (fitxer log), capturar paquet (iniciar captura) o tancar Sessió.
El cas d'us Obrir Captura, inclou Mostrar Dades, ja que si no hi ha fitxer de log no es podem mostrar cap data, també inclou Estadístiques, per el mateix motiu.
Es cas d'us Capturar Paquet estén Guardar Captura, ja que l'aplicatiu guarda un log amb cada paquet capturat. Naturalment també inclou els casos Mostrar dades i Estadístiques. El cas d'us Triar Interfase va associat a Capturar Paquet, ja que es un requisit per capturar dades.
El cas d'us Tancar Sessió no implica cap altre cas d'us ja que es independent.

2.5 Descripció dels casos d'us

Cas d'us 1: "Triar interfase"

Resum de la funcionalitat: mostra per pantalla les interfases de xarxa trobades dintre l'ordinador i el usuari tria el que vol capturar, i a mes, les opcions de filtratge.
Actors: usuari.
Casos d'us relacionats: Capturar paquets.
Precondició: l'ordinador utilitzat disposa d'alguna targeta de xarxa.
Post condició: la pantalla ens mostra les targetes disponibles.
El usuari tria l' interfase que vol capturar i les opcions de filtratge que vol fer.
Alternatives de procés i excepcions: la absència de cap interfase de xarxa.

Cas d'us 2: "Mostrar dades"

Resum de la funcionalitat: mostra per pantalla la captura de dades.
Actors: usuari**.**
Casos d'us relacionats: Obrir captura, Capturar paquet.
Precondició: el fitxer de captura existeix o i han dades en el buffer de captura.
Post condició: la pantalla ens mostra les dades de la captura formatejada.
L'usuari, obre un fitxer de captura existent o finalitza la captura en curs. El usuari pot veure per pantalla una llista formatejada amb els Camps del paquets capturats.
Alternatives de procés i excepcions: que no trobi el fitxer de captura o que no pugui obrir el buffer.

Cas d'us 3: "Obrir captura"

Resum de la funcionalitat: recupera los dades de una captura ja existent.
Actors: usuari.
Casos d'us relacionats: Mostrar dades.
Precondició: el fitxer de captura existeix.
Post condició: la pantalla ens mostra les dades de la captura formatejada.
L'usuari, obre un fitxer de captura existent.
Alternatives de procés i excepcions: que hi hagin problemes amb l'obertura del fitxer de captura.

Cas de us 4: "Captura de paquets"

Resum de la funcionalitat: obre una sessió de captura de paquets que circulen per la targeta de xarxa, que l'usuari a triat prèviament, i filtre segons les opcions que l'usuari també ha triat.
Actors: usuari.
Casos d'us relacionats: Guardar captura, Mostrar dades, Triar interfase.
Precondició: el usuari ha triat l' interfase i les opcions de filtratge.
Post condició: obtenir una captura per a poder veure-la per pantalla i/o guardar-la a un fitxer.
El usuari ha triat l' interfase del que vol obtenir la captura, així com les opcions de filtratge. El procés va mostrant per pantalla els paquets seleccionats, fins que l'usuari pitgi el botó "fi". Una vegada pitjat el botó pot triar entre guardar la captura a un fitxer o descartar-lo.
Alternatives de procés i excepcions: Que hi hagi qualque problema per a crear el fitxer, o que el buffer estigui buit.

Cas d'us 5: "Tancar Sessió"

Resum de la funcionalitat: Restaura la pantalla, a l'estat inicial. Esborra totes les taules.
Actors: usuari.
Casos d'us relacionats: Captura de paquets, estadístiques.
Precondició: Cap.
Postcondició: Tindrem la aplicació a punt per iniciar un altre sessió de captura, sense sortir del programa.
Alternatives de procés i excepcions: Cap.

Cas d'us 6: "Estadístiques"

Resum de la funcionalitat: Mostra per pantalla una seria d'estadístiques en forma de gràfics de la sessió de captura finalitzada.
Actors: usuari.
Casos d'us relacionats: Guardar captura.
Precondició: Existeix el fitxer de captura.
Postcondició: Mostrar en pantalla un resum gràfic de la sessió de captura.
Alternatives de procés i excepcions: Que hi hagi qualque problema per a obrir el fitxer.

2.7 Requisits de la interfície i descripció del casos d'us.

La interfície serà de forma totalment gràfica i intuïtiva, el programa presentarà les diferents opcions mitjançant botons de tria y les opcions de filtratge es trobaran a un camp de text, per que el usuari pugui filtrar segons les convencions de tcpdump, que es una utilitat que ve incorporada als sistemes operatius UNIX: Linux, Solaris, BSD, Mac OS X, HP-UX y AIX, i a Windows (amb aquest cas es diu WinDump).

La utilització dels filtres es la següent:

- type [host|net|port]: Màquina en particular [host], xarxa complerta [net] o port concret [port].
- dir [src|dst|src or dst|src and dst]: Especifica des de [src] o on [dst] es dirigeix la informació.
- proto [tcp|udp|ip|ether]: Protocol que volem capturar.

Cas d'us 1: "Triar interfase"
La aplicació detecta quines interfícies de xarxa estan disponibles en el ordenador.
El usuari tria quin interfase vol capturar.
El usuari tria quines opcions de filtratge vol utilitzar per la captura.

Cas d'us 2: "Mostrar dades"
Un cop que el usuari dona per finalitzada la captura, o be, tria un fitxer de captura anterior, el sistema mostra en forma de taula, les dades mes significatives de cada paquet capturat.
Si el usuari fa click damunt un paquet, se li obri una finestra on pot veure informació detallada del paquet triat.

Cas d'us 3: "Obrir captura"
El usuari tria la opció de obrir fitxer, tot seguit el sistema li mostra una pantalla de navegació, on es veuen tots el fitxers guardats. Fent un click damunt el fitxer, el programa llegeix aquest y el col·loca dintre de la memòria per a visualitzar-lo.

Cas de us 4: "Captura de paquets"
Una vegada que el usuari prem el boto start, el sistema comença la captura de paquets de l'interficie seleccionat i les condicions de filtratge, i va mostrant per pantalla els paquets identificats.
La captura finalitza quant l'usuari prem el boto stop, o quant s'arriba el nombre màxim de paquets dintre el buffer.

Cas d'us 5: "Tancar sessió"
El usuari tria la opció de tancar sessió. El programa fa netes totes les taules u variables utilitzades a la sessió en curs. Prepara la pantalla al estat inicial per iniciar una nova sessió de captura..

Cas d'us 6: "Estadístiques"
Si el usuari tria l'opció estadístiques, el programa obre una nova finestra, fent un resum dels paquets capturats per tipus, així com una representació gràfica d'aquests.

2.8 Disseny del projecte

2.6.1 Disseny de l'interficie d'usuari

L'interficie d'usuari recull tots el cassos d'us que es definiren a la tasca 2.
De forma gràfica la pantalla inicial seria la següent:

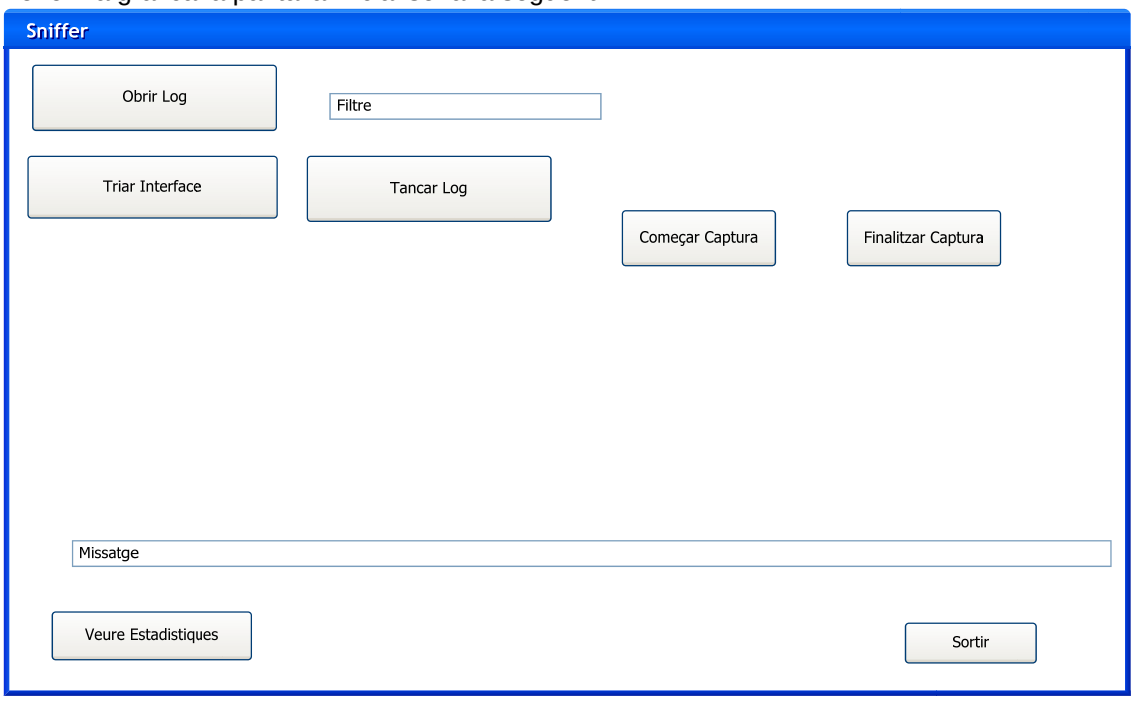

Figura 19 Pantalla d'inici

El botó Triar Interfase ens duria a la pantalla següent:

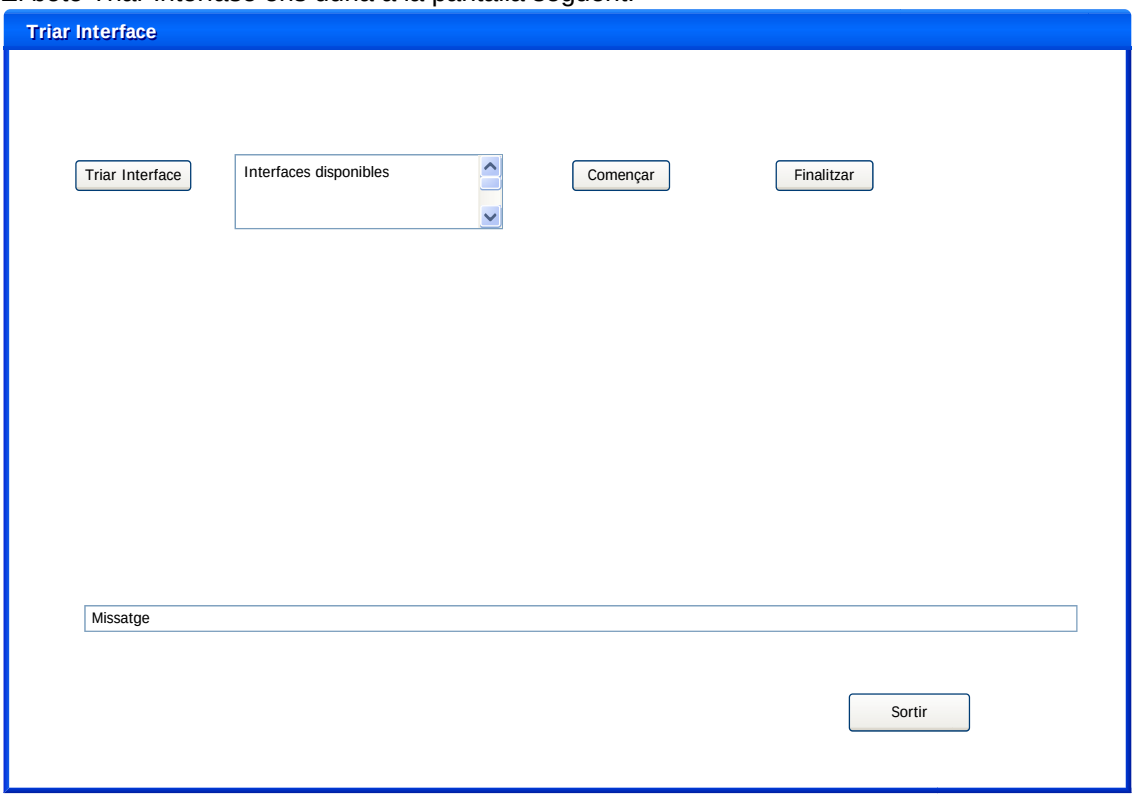

Figura 20 Triar Interfase

El botó Obrir Log, presentarà aquest formulari.
Des de aquí l'usuari Tria el fitxer a obrir i passa a la pantalla següent:

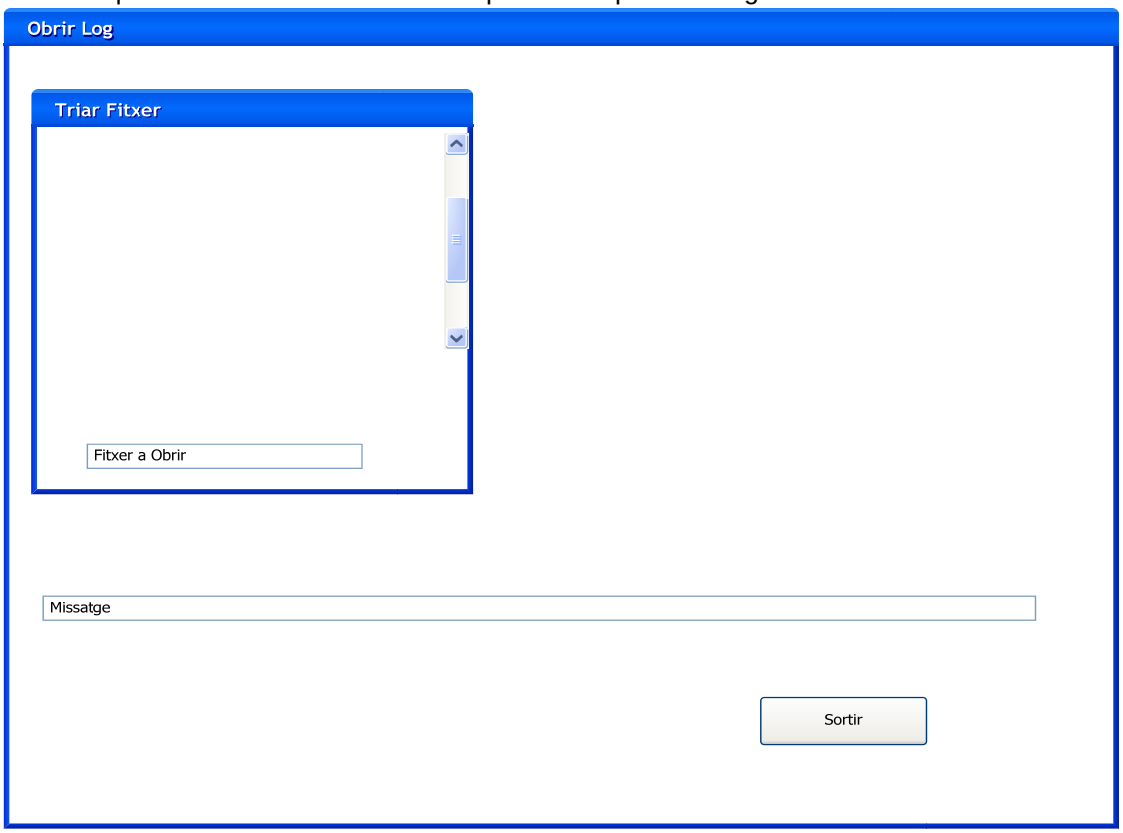

Figura 21 Obrir Fitxer de Log

Des de aquí l'usuari Tria el paquet a expandir i pot recórrer tot el fitxer de log.

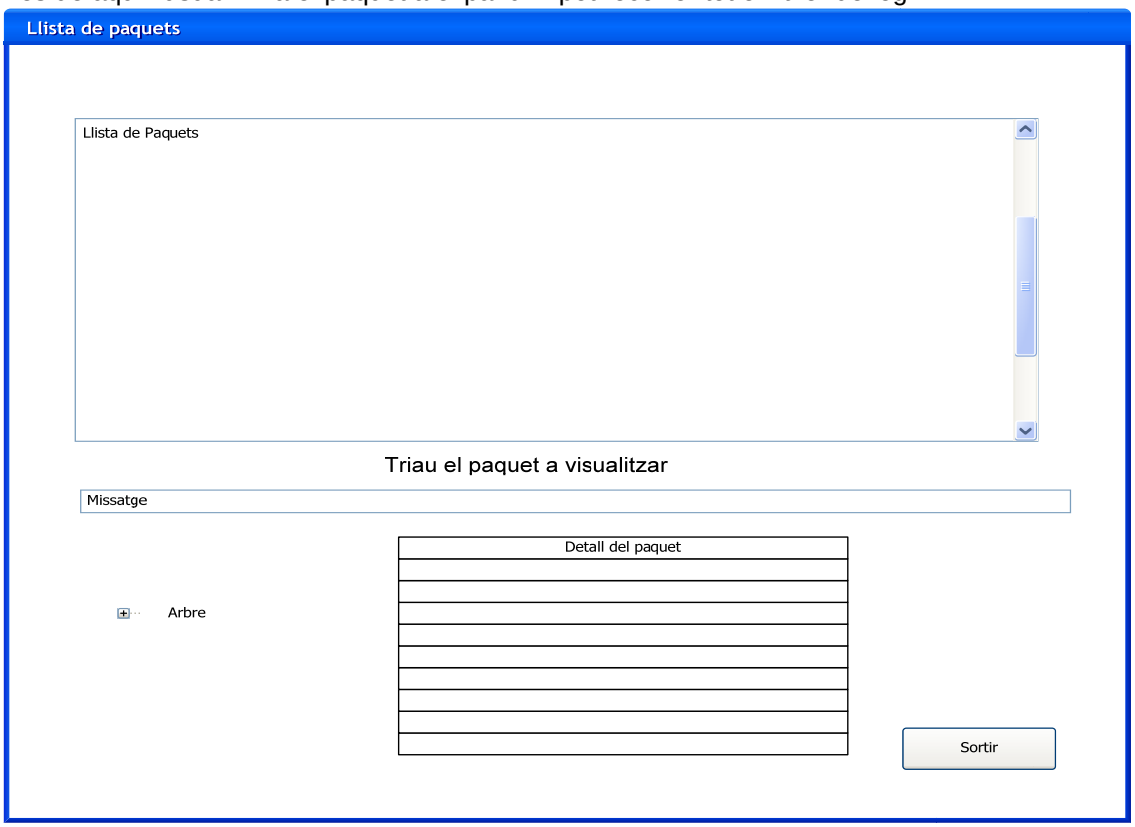

Figura 22 Captura

Aquesta seria l'opció de veure estadístiques

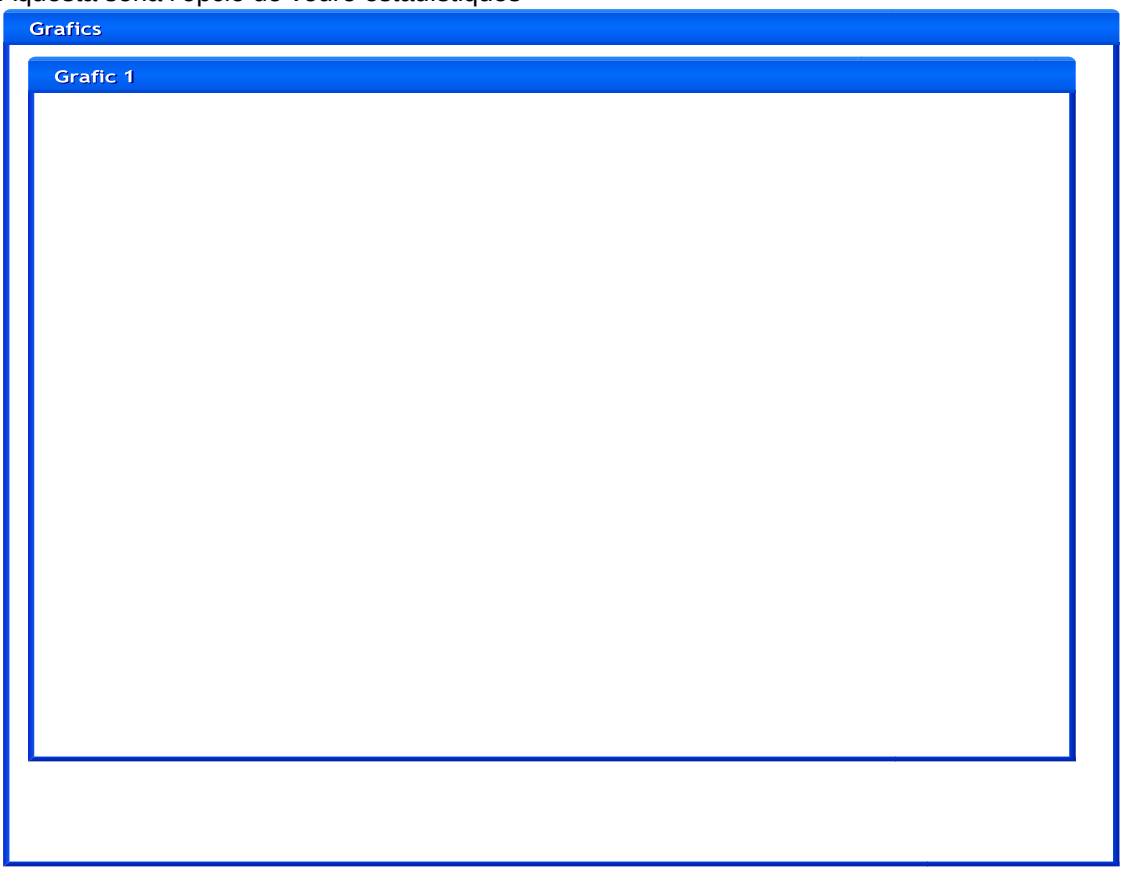

Figura 23 Gràfiques

3. IMPLEMENTACIÓ DEL PROJECTE

3.1 Llenguatge de programació i entorn.

En quant al llenguatge de programació, tenia clar que havia de esser JAVA, donat que el projecte es desenvolupa amb la tècnica d'orientació a objectes, el llenguatge mes estes per aquest sistema es JAVA, tot i que donat la meva experiència professional, sempre ha estat lligada a la programació estructurada, la formació rebuda durant la carrera a la UOC ha estat la de programació orientada a objecte.

Per jo representa un repte molt fort, ja que la experiència amb llenguatges orientats a objecte es molt petita, i si vens del mon de la programació estructurada, et costa molt adapta el teu pensament en el mon dels objectes (sempre et queda la estructura mental del altre tipus de programació).

La següent tasca era decidir un entorn de treball (IDE), per treure un rendiment acceptable amb el poc temps que tenia, tenia clar que nos es molt efectiu programar amb Java amb un editor de texts (tot i que es possible).

Durant la carrera em fet feina amb l'Eclipse, però donat que el projecte requeria una interfase gràfica, vaig veure que el IDE Netbeans, ja duia incorporada un eina per dissenyar components gràfics (pantalles, botons, scrolls, taules, finestres, etc.), que em podia estalviar un temps preciós amb la tasca de definir aquets components. Aquesta eina es Swing, i es tracta de un editor gràfic, que et permet col·locar i dissenyar components en temps real, a més et permet definir accions (*events*) associats a cada component.

3.2 Llibreries necessàries.

Donat que Java, te el seu abast una gran quantitat de llibreries, d'us gratuït, la feina es trobar les mes adients per el projecte en curs.

Per tant l'altre tasca que vaig assolir es cercar alguna llibreria que hem dones un cop de ma, per capturar trames de xarxa, durant el curs de la assignatura de xarxes, varem programar sockets, però això es molt farragos. Vaig trobar una llibreria (JPCAP), d'us lliure que te tots el mètodes necessaris per a capturar i interpretar els paquets que viatgen per les xarxes.

L'altre eina que necessitava era alguna llibreria per a fer gràfiques, la triada va esser Jfreechart, tot i que molt mes del que necessitava, la vaig trobar molt senzilla d'utilitzar.

3.3 Entorn de desenvolupament

El IDE NETBEANS, es molt fàcil d'instal·lar, nomes s'ha de descarregar de la URL,**http://netbeans.org/kb/index.html,** i seguir les instruccions, disposa d'un tutorial en línea, per aprendre a utilitzar-lo, i nombrosos exemples del que podem fer.
També hem de instal·lar el Java JDK, i la màquina virtual de JAVA (JRE). Concretament per aquest projecte s'ha utilitzat el Netbeans 7.2.1 , el JDK 1.7 i el JRE 7.

Instal·lació de les llibreries necessaris.

Primer la llibreria Jpcap, que incorpora els mètodes necessaris per el tractament de paquets de xarxa.

http://netresearch.ics.uci.edu/kfujii/Jpcap/doc/tutorial/index.html
Aquesta llibreria s'integra perfectament amb el IDE NETBEANS.

http://sourceforge.net/projects/jpcap/

Figura 24 Documentació de jcap

També s'ha hagut d'instal·lar la llibre JFreeChart, per fer el gràfics.

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

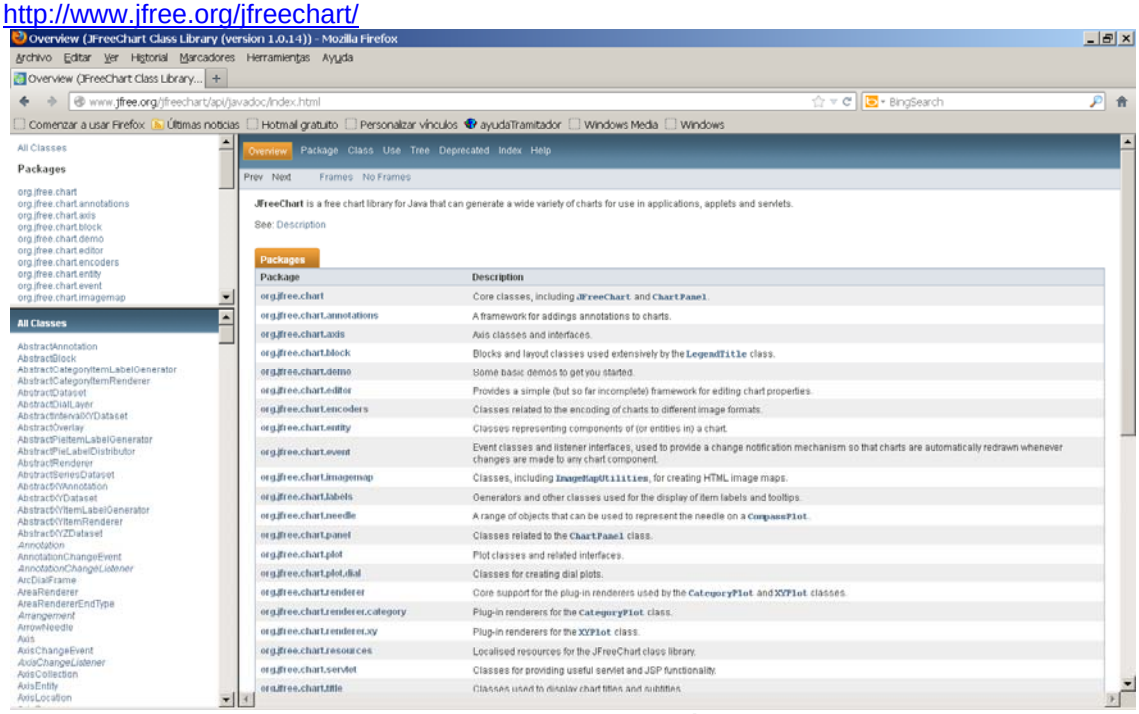

Figura 25 Documentació de jFreechart

Aquí també es troben tots el mètodes necessaris per fer qualsevol tipus de gràfic.

A mes de tot això, he incorporat al NetBeans, el plugin JAR Maker, el qual ens permet obtenir un arxiu tipus JAR, per a poder distribuir la aplicació a qualsevol usuari, i que a més es independent de la plataforma, ja que es pot executar en qualsevol entorn que disposi de una maquina virtual JAVA.

3.4 Consideracions a l' implementació.

La captura de paquets amb thread.(fil).

He decidit utilitzar el threads (fils), per a poder controlar la finalització de les captures, ja que des de dintre de la classe principal, es feia gaire be impossible controlar la finalització de la captura.
La utilització del threads amb Java es bastant senzilla,
Quant s'inicia un programa en Java, aquest té un fil: el fil principal. Podem interaccionar amb el fil principal de diferents maneres, obtenint o assignant el seu nom, aturant-lo i d'altres funcionalitats més. No obstant, també podem iniciar d'altres fils. Podem iniciar el codi d'aquests fils amb una crida al mètode start de l'objecte i col·locant el codi que volem executar en el mètode run.
De fet he creat una classe FerCaptura, que estén la classe Thread, i aquest fil es utilitzat tant per capturar paquets de l'interface de xarxa com des de el fitxer de log.
Per aturar el fil, nomes hem de cridar el mètode stop de l'objecte.
El problema mes greu ha estat com passar l' informació del fil a la classe principal, axó s'ha resolt passant com a paràmetre al fil una instancia de la classe principal (NSN_SniffferUI), i fent que el fil utilitzi el mètodes de la classe principal per omplir la Jtable que s'utilitza per visualitzar els paquets capturats.

El filtratge dels paquets.

Per a filtrar els paquets, s'ha incorporat un camp de text, el qual si està a null no s'aplica cap filtre, i si te contingut, s'ha aplica aquest contingut com a filtre. El problema es que no es fa cap depuració del format del filtre, per tant si l' informació introduïda no es bona, el programa provoca una excepció.
El format de filtra acceptat es el que segueix les convencions de tcpdump, que es una utilitat que ve incorporada als sistemes operatius UNIX: Linux, Solaris, BSD, Mac OS X, HP-UX y AIX, i a Windows (amb aquest cas es diu WinDump).

La utilització dels filtres es la següent:

- **type** [host|net|port]: Màquina en particular [host], red complerta [net] o port concret [port].
- **dir** [src|dst|src or dst|src and dst]: Especifica des de [src] o on [dst] es dirigeix la informació.
- **proto** [tcp|udp|ip|ether]: Protocol que volem capturar.

Exemples:

tcp or udp (filtre nomes els paques tcp o udp).
src host 192.168.3.1 (captura els Parquets amb origen la ip 192.168.3.1
host 192.168.3.2 (filtre només els paquets amb origen o destí la ip 192.168.3.2
dst port 23 (filtre els paquet amb port de destí 23)

La visualització de l' informació dels paquets.

Els paquets capturats, es visualitzen en forma de taula , amb informació mínima de cada paquet (nro. de captura, ip origen, ip desti, tipus de paquet, paquet).
Una vegada que el usuari tria una fila de la taula s'expandeix la informació del paquet amb dues parts**.**

La visualització de la informació dels paquets es fa amb dos objectes, el primer es un Tree (arbre), on es troben desglossades jeràrquicament, la informació de cada una de les capes de un paquet. L'altre es una caixa de text, on se mostra en format hexadecimal el contingut complert del paquet.

La part de l'arbre es la m'ha donat mes feina, ja que la llibreria Jpcap incorpora mètodes per a veure les parts principals de un paquet, però aquest detalls s'han de incorporar a un objecte de tipus Tree, per a formatejar la informació de les diferents capes de un paquet, ja que la meva intenció era la de donar una informació el mes amplia possible de la informació de les trames, he hagut de crear una classe *AnalitzaPaquets*, amb els mètodes necessaris per a esbrinar el contingut de cada un dels camps dels paquets, a aquesta classe se li passa a cada mètode com a paràmetre un objecte de tipus paquet, del tipus que indica el seu contingut (el qual s'ha detectat i emmagatzemat a la captura dins la classe *EstructuraPaquet*, amb una instancia de la subclasse que ens proporciona Jpcap on estan definits el mètodes per cada tipus de paquet. Es fa una cridada al mètodes d'aquesta classe per a cada un dels nodes del arbre. Aquests tornen un String o un Array, amb la informació a punt per esser col·locada al arbre.

Primer es defineix una nova instancia del arbre per a cada paquet de la forma:

```
// Per a tots els paquets s'ha de crear la Arrel, el node Frame i el node Ethernet
      DefaultMutableTreeNode root = new DefaultMutableTreeNode("Root");
      DefaultMutableTreeNode frame = new DefaultMutableTreeNode("Frame " +          captura.getNro_paquete()  +
" [" + p.caplen + " bytes]");
      DefaultMutableTreeNode eth = new DefaultMutableTreeNode("Ethernet");
```

Després es crida als mètodes adients de la classe *AnalitzaPaquets*, per omplir cada branca del arbre i les seves fulles.

Per exemple per el primer nivell (capa Frame), es crida al mètode:

String frame_details[] = AnalitzaPaquets.analitza_frame(captura);

Per el segon nivell (capa Ethernet), es crida al mètode.

String eth_details[] = AnalitzaPaquets.analitza_eth((EthernetPacket) p.datalink, p);

Amb l'informació retornada es van omplint les branques comuns.

```
root.add(frame);
root.add(eth);
```

Després per cada tipus de protocol es crida el mètode adient, per a omplir la resta.

```
      root.add(ip);
      root.add(tcp);  // en cas de protocol TCP
```

Per cada un dels protocols reconeguts, es va construint el arbre jeràrquic.

L'altre part de la informació del paquet, es mes senzilla, es mostra a una caixa de text tot el contingut el paquet en format hexadecimal.

Obrir el fitxer de log.

Per aquesta tasca s'utilitza el component Swing objecte tipus fileChooser amb el qual es pot navegar i que ens retornarà el nom del fitxer a recuperar. Per a recuperar les dades, s'utilitza el mateix thread que per la captura d'interfase, nomes que amb passant-li un flag perquè no guardi res a cap fitxer de log.

Les estadístiques.

Aquí es on s'utilitza la llibreria Jfreehcart, el programa comprova si hi han dades per mostrar, i si es així, calcula les captures per cada tipus de paquet i li passa les dades al mètode de jfreechart per a construir la gràfica.

```
JFreeChart chart = ChartFactory.createBarChart3D(
            ("Paquets capturats per protocol"), // Títol del Gràfic
            "", // Etiqueta Eix X
            ("Protocols"), // Etiqueta Eix Y
            dataset, // Dades
            PlotOrientation.VERTICAL,
            true, // Incluir legenda
            true, // tooltips
            false // URLS
            );
```

Tancar Sessió.

Aquest botó s'ha incorporat degut a la necessitat de fer net la finestra, quant volem fer una altre acció, però no volem sortir del programa.
El que fa es buidar el contingut dels objectes i deixar-lo a l'estat inicial.

4.PROVES

Resum de proves efectuades

Filtrat per tipus tcp o arp. Aquí es pretén filtrar nomes els paquets de tipus ARP o TCP.
Es pot veure que a la columna de Tipus només apareixen els paquets del tipus desitjat.

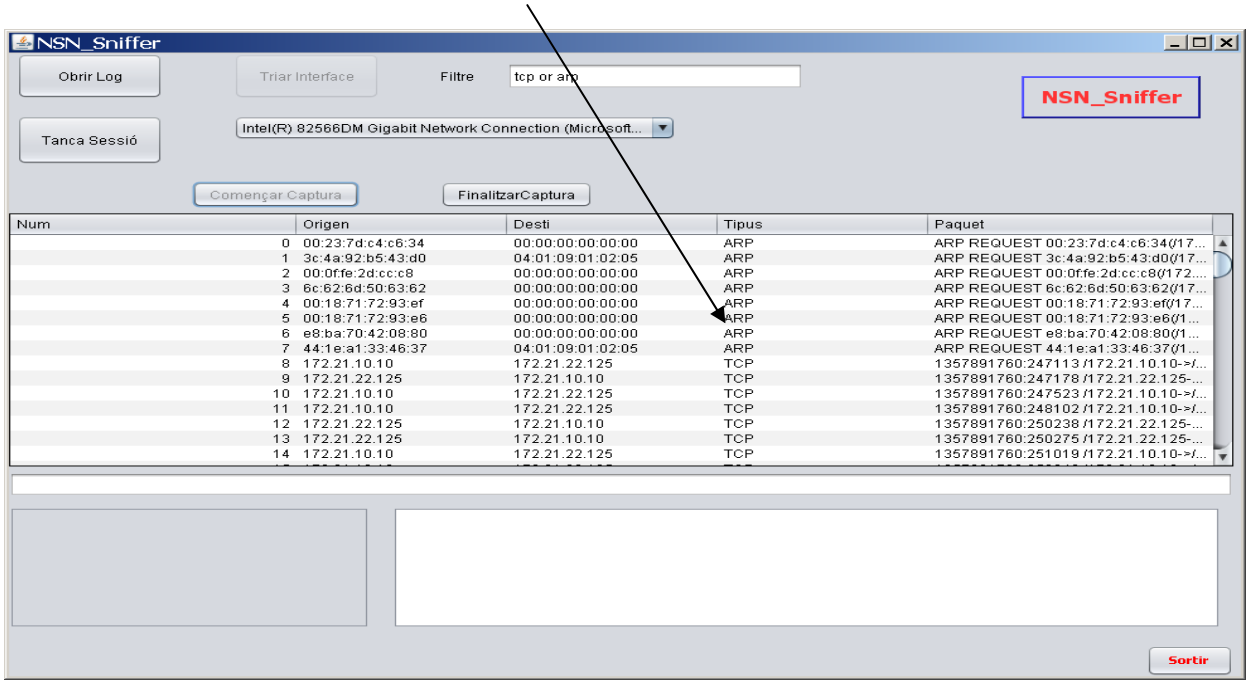

Figura 26 Filtrat per tipus

Filtrat per tipus src host (tots el paquets que tinguin com inici la ip 172.21.22.125.
Es pot veure a la columna Origen com nomes es capturen paquets amb origen la ip desitjada.

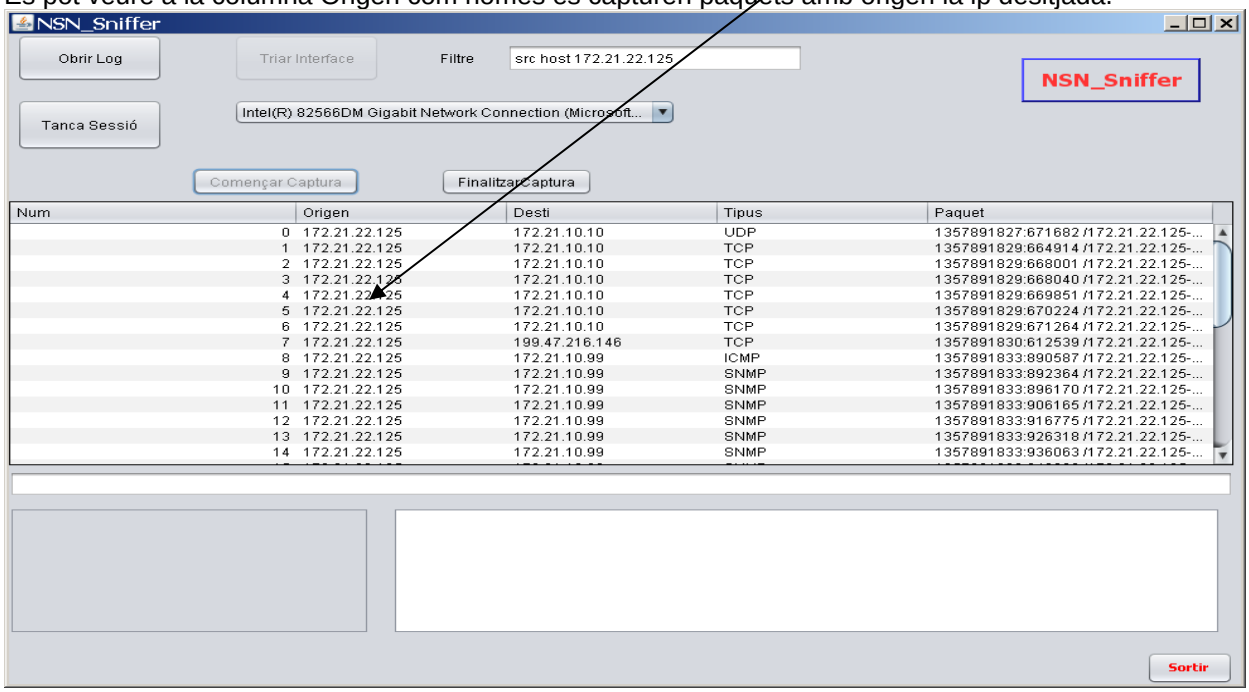

Figura 27 Filtrat per src host

Filtrat per tipus host (tots el paquets que tinguin com inici o destí la ip 172.21.22.125.
En aquest cas a les columnes Origen i Destí ens captura els paquets que tinguin com Origen o Destí la IP triada.

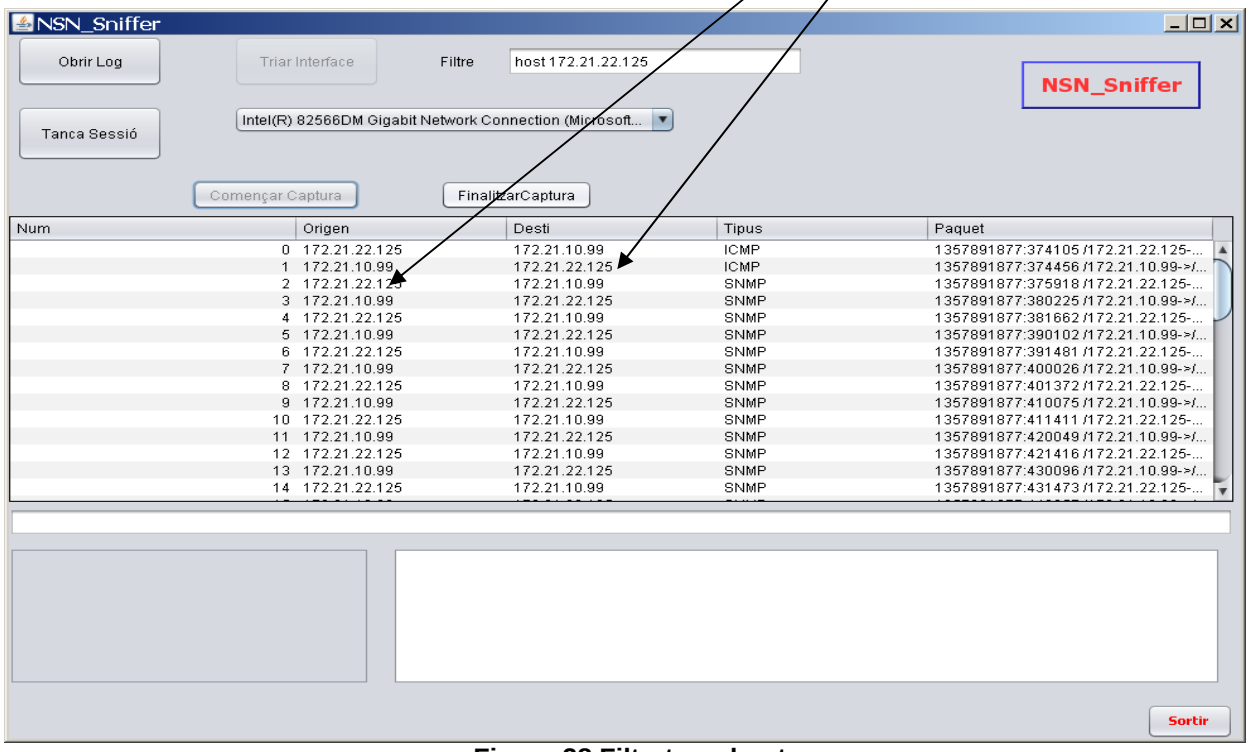

Figura 28 Filtrat per host

Gràfiques per tipus de paquet capturat

En aquesta captura es veuen el nombre de paquets capturats per tipus.

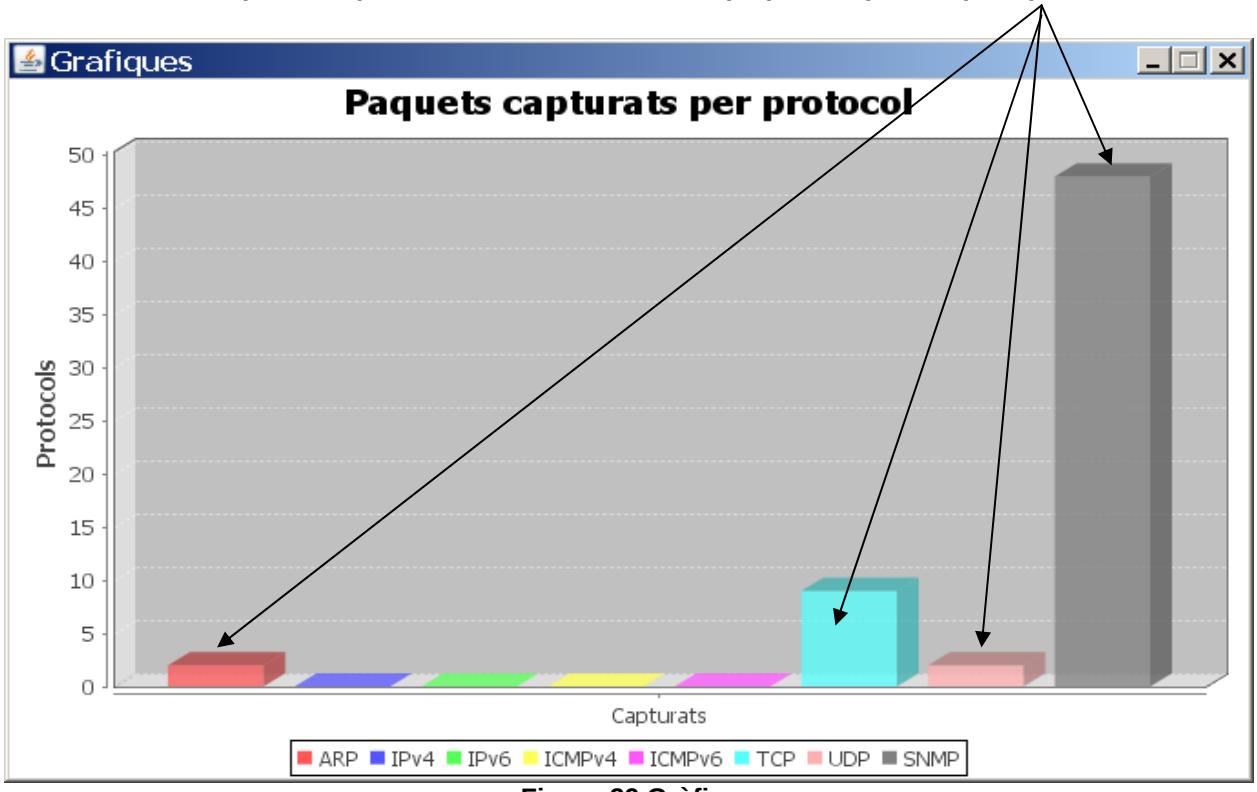

Figura 29 Gràfica

Captura des de log i expansió de dades de paquet

Podem veure que s'ha seleccionat el paquet numero 4 i ens obren les finestres d'expansió de tipus arbre i text hexadecimal amb el contingut del paquet seleccionat.

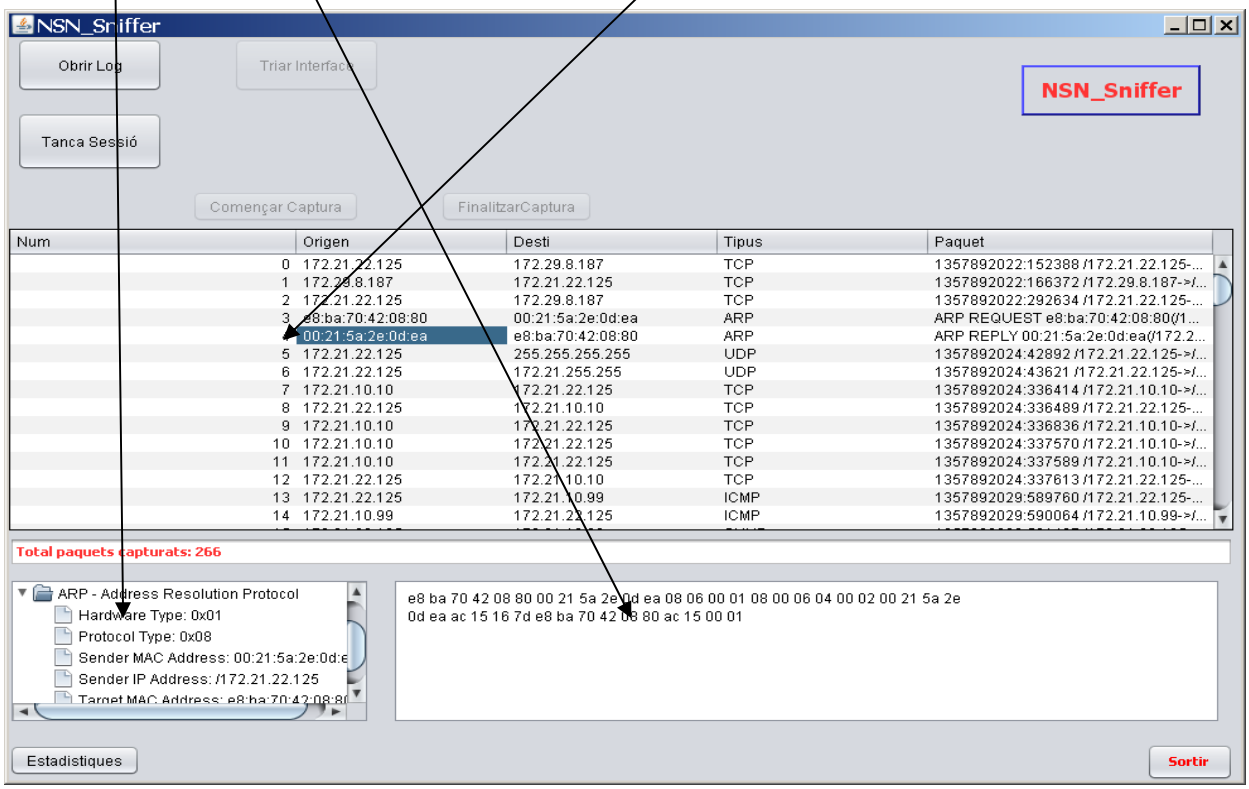

Figura 30 Pantalla de captura de log

5. MANUAL D'INSTAL·LACIÓ I D'USUARI

Aquest programa ha estat desenvolupat amb un equip Intel QuadCore 2600 , amb sistema operatiu Windows 7, 4 Gb de RAM i dues targes Ethernet, una connectada a un router, i l'altra Wifi també connectada a un router. L'entorn de desenvolupament utilitzat ha estat Netbeans amb la versió 7.2.1.

5.1 Manual de la instal·lació

Requeriments:
Un equip amb una màquina virtual JAVA (jre 1.6 or superior).
Sistema operatiu Windows XP o superior o Linux.
No necessita res més si s'executa el JAR.
Si es vol executar des de un IDE amb Windows, necessita tenir instal·lada la llibreria jpcap i abans d'aquesta la llibreria winpcap.

5.1.1 Instal·lació en sistemes Windows.

Per a instal·lar winpcap
La llibreria winpcap es pot baixar de l'adreça electrònica http://www.winpcap.org/
A partir d'aquí només haurem de fer doble click per a instal·lar-la.
Per a instal·lar jpcap
La llibreria jpcap la pots baixa de la següent adreça electrònica
http://sourceforge.net/projects/jpcap/

1. Descarregar i extreure el codi font de Jpcap
2. Copiar "lib/Jpcap.dll" al directori [JRE\bin]
3. Copiar "lib/jpcap.jar" en el directori [JRE\lib\ext]
4. Si es vol instal·lar JDK, també necessita copiar "lib/jpcap.jar" en el directori [JDK\jre\lib\ext]

Configurar les variables d'entorn, que son les que informen al Sistema Operatiu on i com ubicar Java dintre el mateix, aquestes variables son : "JAVA_HOME" y "PATH".

- Per JAVA_HOME s'ha de definir on es troba el JDK, normalment a c:\program Files\Java\jdk
- Per la variable PATH s'ha d'afegir a la variable PATH existent %JAVA_HOME%\bin per exemple C:\WINDOWS;C:\WINDOWS\system32;%JAVA_HOME%\bin

Copiar els arxius continguts a la carpeta \dist\ en un directori que haurem creat a tal fi.
Per a executar el programa haureu de seleccionar amb el botó de la dreta el fitxer "NSN_Sniffer.jar" i escollir l'opció "Abrir con-> javaw".

Nota: el directori JRE normalment es troba a "C:\Archivos de Programa\Java\jre" i el directori JDK habitualment es troba a "C:\Archivos de Programa\Java\jdk". En cas de que la màquina utilitzi un sistema Windows de 64 bits, els directores mencionats es troben a C:\Arhivos de Programa(x86)\

5.1.2 Instal·lació en sistemes Linux.

Primer hem de copiar el *.jar a la carpeta on el vulguem tenir.
Per aconseguir fer funcionar l'aplicació amb un entorn Linux, s'han de complir dues condicions:

1. Tenir instal·lada la màquina virtual de Java
2. Definir el programa a utilitzar per obrir els fitxers .jar

El JRE o Java Runtime Enviroment no es mes que el conjunt d' aplicacions y llibreries necessàries para poder utilitzar una aplicació Java
A continuació es mostra un exemple amb una distribució Linux Ubuntu.
Definir el programa per obrir programas Java
Basta fer click amb el boto dret del mouse sobre el fitxer *.jar* que conte l'aplicació i seleccionar l'opció que apareix en la imatge

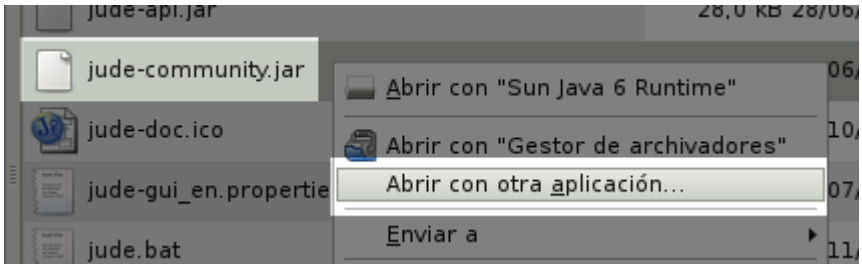

Figura 31 Obrir amb Linux 1

En el formulari que ens apareix a continuació escrivim java –jar tal y com es pot observar en la figura

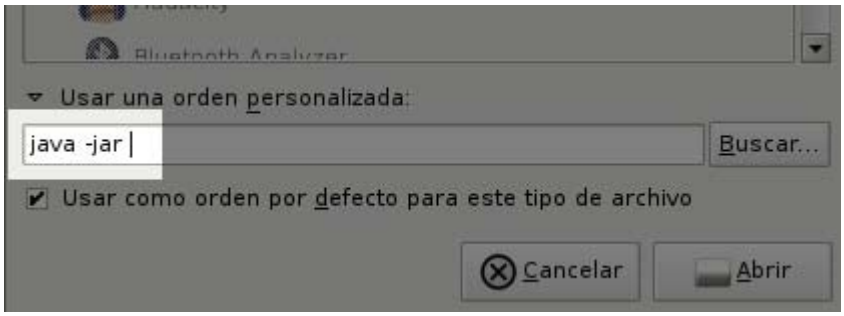

Figura 32 Obrir amb Linux 2

NOTA: Darrera el *-jar* es aconsellable deixar un espai en blanc per evitar problemes.

5.2 Manual d'usuari.

Una vegada engegat el programa ens surt la pantalla següent des de on podem governar tota la mecànica del programa.

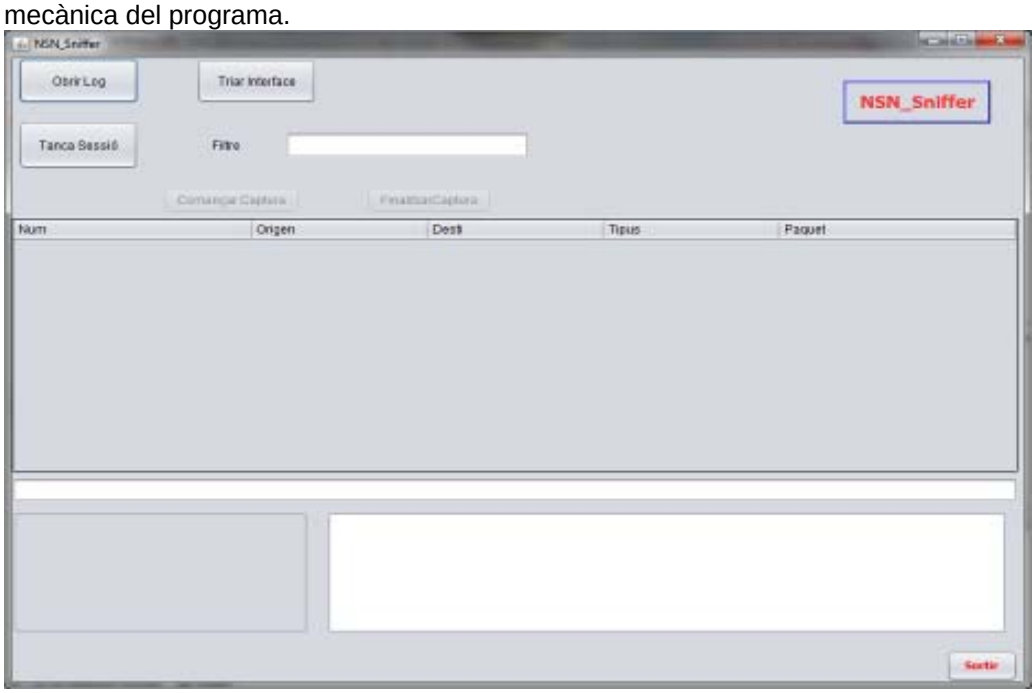

Figura 33 Pantalla d'inici

Aquí, tenim un boto per a cada opció o cas d'us.
Per exemple si volem fer una captura, el primer que em de fer es triar un interfase des de on volem capturar, i omplir el camp de filtre si es cal.

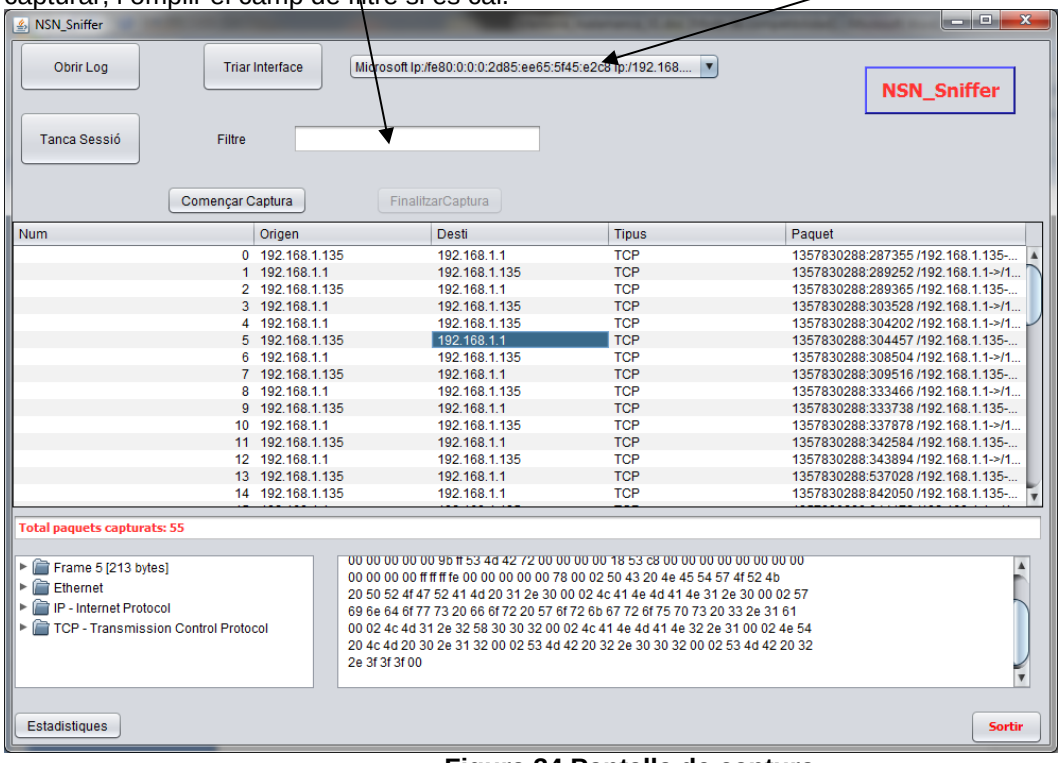

Figura 34 Pantalla de captura

Una vegada triat, el 'interfase desitjat, del qual ens surt la descripció i la ip, si la te, es quant podem introduir el filtre que volem, i pitjar el boto **començar captura** per iniciar la captura.
El procés de captura ens va mostrant a la taula principal els paquets que es van capturant, fins que pitgem el botó **finalitzar captura**.

Una vegada pitjat el botó **finalitzar captura**, podem clicar a qualsevol fila, i ens mostra les finestres de detall del contingut del paquet. La primera en forma de arbre, el qual podem estendre faent click a cada una de les branques, i la segona a la dreta el contingut del paquet en format hexadecimal.

Si en aquets moment volem veure les estadístiques de forma gràfica, podem pitjar el boto **Estadístiques** i ens mostra la següent pantalla.

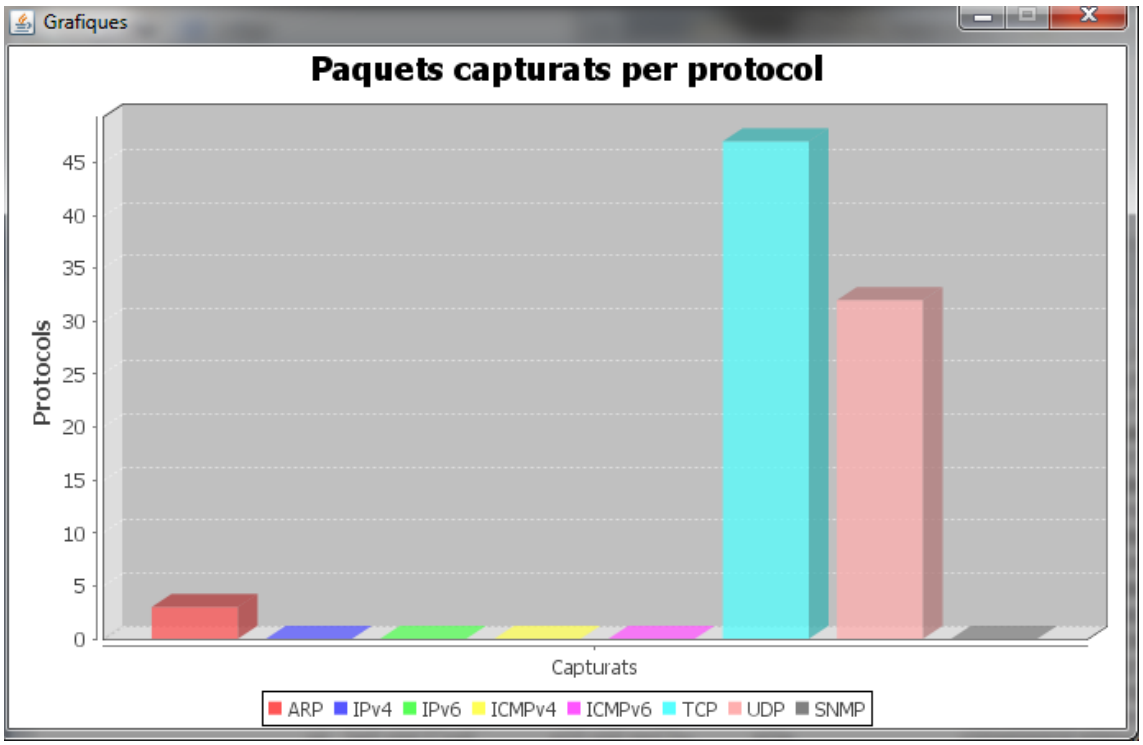

Figura 35 Pantalla de gràfiques

6.CONCLUSIONS

Como conclusió final, he de dir que amb un principi pareixia un projecte senzill, quant he estat ficat dintre em semblava que no ho podria acabar, i una vegada acabat sembla un altre vegada senzill.

Aquest raonament s'ha d'explicar un poc. Em semblava senzill en un principi, perquè tenia molt clar el que havia de fer el programa, es a dir els casos d'us estaven molt clars, i no semblava molt difícil d'implementar, el problema en el seu inici podia esser com fer per manejar un interfase de xarxa i com a representar les gràfiques, però una vegada trobades les llibreries adients, seguia semblant senzill.

Una vegada ficats en feina, els problemes han estat nombrosos, sobre tot per la falta de experiència en Java, durant la carrera hem fet algunes practiques de Java, on s'expliquen les bases d'aquest llenguatge, però la seva dificultat no es la sintaxis, sinó entendre la teoria dels objectes, les classes, els mètodes, les herències, el polimorfisme, el threads. En fi aquesta ha estat la gran dificultat.

Quant s'arriba al final, vista la estructura que ha quedat, sembla molt mes senzill, ja que un cop controlades les dificultats de la programació orientada a objectes, es veuen les avantatges que aporta, tot i que te una corba d'aprenentatge molt elevada.

Estic molt content amb la feina perquè ha estat un repàs casi complert per els continguts de bastants assignatures de la carrera, i penso que aporta un assentament dels coneixements adquirits al curs d'aquests anys.

En el meu cas, i donat que tinc una visió global del mon de l'informàtica, que em donen 33 anys de professió, i que he tingut la sort de viure amb primera persona els canvis que s'han assolit, (vaig començar amb un NCR Century 101 amb cinta perforada i memori de ferrita, vaig programar els primers Apple II, després vaig assistir al naixement del PC de IBM, vaig tornar al mon del mainframe amb un Sperry Univac S/80, amb un IBM S36 i finalment amb IBM 390, he programat en llenguatges com NEAT III, COBOL 74, ensamblador IBM i INTEL , NATURAL. BASIC, i altres que ja no recordo, per finalment passar a cap de projectes i desconnectar casi del tot amb la programació, fins ara. Per aquest motiu. ha estat si tot mes interessant i gratificant, donat que també ma servit per una posada al dia molt important, i tornar als orígens, quant la meva feina era la programació pura i dura. Desgraciadament, en el mon de l'informàtica, quant la teva feina va essent mes directiva que productiva, es va perdent de vista les dificultats que tenen els programadors i les eines que se els han de proporcionar per fer la seva feina mes productiva i gratificant.

En fi, després d'aquestes conclusions, ens centrem en les possibles millores del NSN_Sniffer.
En mes temps per davant, trobo que li falten les funcionalitats següents:

- Una funció per poder guardar els logs de les capturats amb noms diferents (seria bastant senzill d'implementar).
- Mes informació estadística i gràfica (paquets per ports, per grups d'ips, per temps).
- Una interfase de filtres mes elaborada, que facilites la feina al usuari.
- Una versió per Android, aquesta era la meva intenció inicial, però no hi avia temps per mes.

7. GLOSSARI

ARP (Address Resolution Protocol), Protocol per al mapatge automàtic d'adreces MAC amb adreces IP.
Adreça MAC Identificador hexadecimal de 48 bits que correspon de forma única a una targeta de xarxa.
Adreça IP Nombre que identifica de forma lògica una interfície de xarxa. En la seva versió 4, aquest nombre es representa mitjançant un nombre binari de 32 bits.
ICMP (Internet Control Message Protocol), Protocol de nivell de xarxa emprat
només per a tasques de gestió del nivell de xarxa.
IGMP (Internet Group Management Protocol), Protocol que es utilitzat exclusivament pels membres d'una xarxa multicast per tal de propagar informació de redireccionament.
IP (Internet Protocol) Protocol de nivell de xarxa utilitzat a Internet.
Java Llenguatge de programació orientat a objectes desenvolupat per Sun
Microsystems i que permeten crear programes multiplataforma.
jpcap Llibreria implementada en Java que proporciona un interfície de programació de comunicacions.
MAC (Medium Access Control) Protocol de control d'accés al medi utilitzat per les estacions d'una mateixa xarxa d'àrea local.
Protocol Conjunt de codis i formats que es fan servir perquè els ordinadors puguin intercanviar informació.
RFC (Request For Comment), Cadascun dels documents que expliquen amb tot detall els diferents protocols de la xarxa Internet.
TCP (Transmission Control Protocol), Protocol a nivell de transport que permet
garantir que el intercanvi d'informació s'ha realitzat correctament.
Thread Cadascun dels processos que s'executen en paral·lel a una màquina i que permeten la utilització del processador amb un temps compartit.
UDP (User Datagram Protocol), Protocol a nivell de transport utilitzat quan no es necessari la confirmació d'un intercanvi d'informació
SNMP (Protocol Simple d'Administració de Xarxa o SNMP)(de l'anglès Simple Network Management Protocol) és un protocol de la capa d'aplicació que facilita l'intercanvi d'informació d'administració entre dispositius de xarxa. Permet als administradors supervisar el funcionament de la xarxa, buscar i resoldre els seus problemes, i planejar el seu creixement.
Les versions de SNMP més utilitzades són SNMP versió 1 (SNMPv1) i SNMP versió 2 (SNMPv2).

8. BIBLIOGRAFIA.

- Documentació oficial de la llibreria Jpcap
  http://sourceforge.net/projects/jpcap/

- Programes similars**:**

  WIRESHARK (http://www.wireshark.org/)
  Commview de TAMOSOFT http://www.tamos.com/products/commview/
  Ptraf. ftp://iptraf.seul.org/pub/iptraf/

- Exemples varis d'utilització de rutines amb Java
  http://milarde.blogspot.com.es/2012/05/el-lenguaje-de-programacion-java.html

- Documentació oficial de NETBEANS
  http://netbeans.org/

- Documentació oficial de la llibreria jfreechart.
  http://www.jfree.org/jfreechart/

- Exemples de Jfreechart.

  http://chuwiki.chuidiang.org/index.php?title=Ejemplo_sencillo_con_JFreeChartArea%2C_Bar_y_Line

- Wikipedia. Informació sobre protocols de xarxa
  http://es.wikipedia.org/wiki/Categor%C3%ADa:Protocolos

- Xarxes. Aplicacions i protocols d'Internet
  Jordi Iñigo Griera (coordinador) (UOC). Primera edició: setembre 2003

- Enginyeria del programari.
  -Benet Campderrich Falgueras UOC P01/75007/00565

- Programació orientada a objectes.
  Joan Arnedo Moreno
  Daniel Riera i Terrén (coordinadors) UOC XP07/75063/00241

- Piensa_en_Java_(Bruce_Eckel)
  Segunda_Edición_-_Prentice_Hall

10- ANEXES

Annex 1. Descripció de formats de paquets.

Format de una trama Ethernet

64 bits	48 bits	48 bits	16 bits	**Entre 46 i 1500 bytes**	32 bits
Preàmbul	destí	origen	Tipus	**Dades**	CRC

Preàmbul: Esta format per 64 bits, alternativament 0 y 1 i els dos darrers son 11, permet la sincronització entre els nodes.
Direcció de destí: Direcció MAC de destí.
Direcció d' origen: Ens dona la direcció física (MAC), del transmissor de la trama.
Tipus: Indica el tipus de contingut del camp de dades que porta la trama.
Dades: La seva longitud ha de esser múltiple de 8 bits, es a dir Ethernet transmet la informació en bytes; la seva longitud màxima (MTU)es de 1500 bytes; y a mes el camp de dades ha de tenir com a mínim 46 bytes de llargària.
CRC: Es el codi de redundància cíclica utilitzat per a la detecció de errors y controla tota la trama menys el preàmbul.

Format de un paquet IP.

Versió	Longitud de capçalera	Tipus de serv	Longitud total del paquet	Identificació del paquet	Indicadors	Posició d'aquest fragme	Temps de vida TTL	Protocol	Checksum	@origen	@destí	Opcions
4	4 bits		16 bits	16 bits	3 bits	13	8	8 bits	16 bits	32 bits	32	<=40

Versió: 4 bits
Sempre val el mateix (0100). Aquest camp descriu el format de la capçalera utilitzada. A la taula es descriu la versió 4.
Mida Capçalera (IHL): 4 bits
Longitud de la capçalera, en paraules de 32 bits. El seu valor mínim és de 5 per a una capçalera correcta, i el màxim de 15.
Tipus de Servei: 8 bits
Indica una sèrie de paràmetres sobre la qualitat de servei desitjada durant el trànsit per una xarxa. Algunes xarxes ofereixen prioritats de serveis, considerant determinat tipus de paquets "més importants" que altres (en particular aquestes xarxes només admeten els paquets amb prioritat alta en moments de sobrecàrrega). Aquests 8 bits s'agrupen de la següent manera. Els 5 bits de menys pes són independents i indiquen característiques del servei:

Bit 0: sense ús, ha de romandre en 0.
Bit 1: 1 cost mínim, 0 cost normal.
Bit 2: 1 màxima fiabilitat, 0 fiabilitat normal.
Bit 3: 1 màxim rendiment, 0 rendiment normal.
Bit 4: 1 mínim retard, 0 retard normal.

Els 3 bits restants estan relacionades amb la precedència dels missatges, un indicador adjunt que indica el nivell d'urgència basat en el sistema militar de precedència (vegeu Message Precedence) de l'CCEB, un organització de comunicacions electròniques militars formada per 5 nacions. La urgència que aquests estats representen augmenta a mesura que el nombre format per aquests 3 bits ho fa, i responen als següents noms.

000: De rutina.
001: Prioritari.

010: Immediat.
011: Llampec.
100: Invalidació llampec.
101: Processant anomenada crítica i d'emergència.
110: Control de treball d'Internet.
111: Control de xarxa.

Longitud Total: 16 bits

És la mida total, en octets, del datagrama, incloent la mida de la capçalera i el de les dades. La grandària mínima dels datagrames usats normalment és de 576 octets (64 de capçaleres i 512 de dades). Una màquina no podeu enviar datagrames menors o majors d'aquesta mida a menys que tingui la certesa que seran acceptats per la màquina destí.

En cas de fragmentació aquest camp contindrà la grandària del fragment no el del datagrama original.

Identificador: 16 bits

Identificador únic del paquet. S'utilitzarà, en cas que el datagrama hagi de ser fragmentat, per poder distingir els fragments d'un datagrama dels d'un altre. El originador del datagrama ha d'assegurar un valor únic per a la parella origen destinació i el tipus de protocol durant el temps que el datagrama pugui estar actiu a la xarxa. El valor assignat en aquest camp ha d'anar en format de xarxa.

Indicadors: 3 bits

Actualment utilitzat només per especificar valors relatius a la fragmentació de paquets:

bit 2: Reservat; ha de ser 0
bit 1: 0 = Divisible, 1 = No Divisible (DF)
bit 0: 0 = Últim Fragment, 1 = Fragment Intermedi (li segueixen més fragments) (MF)

La indicació que un paquet és indivisible s'ha de tenir en compte sota qualsevol circumstància. Si el paquet depengui ser fragmentat, no s'enviarà.

Posició de Fragment: 13 bits

En paquets fragmentats indica la posició, en unitats d'64 bits, que ocupa el paquet actual dins del datagrama original. El primer paquet d'una sèrie de fragments contindrà en aquest camp el valor 0.

Temps de Vida (TTL): 8 bits

Indica el màxim nombre d'encaminadors quals alguns paquets travessar. Cada vegada que algun node processa aquest paquet disminueix el seu valor en 1 com a mínim, una unitat. Quan arribi a ser 0, el paquet serà descartat.

Protocol: 8 bits

Indica el protocol de les capes superiors al que s'ha de lliurar el paquet. *Veure Annex 2. Tipus de protocol*

Suma de Control de Capçalera: 16 bits

Es re-calcula cada vegada que algun node canvia algun dels seus camps (per exemple, el Temps de Vida). El mètode de càlcul intencionadament simple consisteix a sumar en complement a 1 cada paraula de 16 bits de la capçalera (considerant valor 0 per al camp de suma de comprovació de capçalera) i fer el complement a 1 del valor resultant.

Adreça IP d'origen: 32 bits S'ha de donar en format de xarxa.

Adreça IP de destinació: 32 bits S'ha de donar en format de xarxa.

Opcions: Variable

Encara que no és obligatòria la utilització d'aquest camp, qualsevol node ha de ser capaç d'interpretar-la. Pot contenir un nombre indeterminat d'opcions, que tindran dos possibles formats:

Format d'opcions simple

Es determina amb un sol octet indicant el Tipus d'opció, el qual està dividit en 3 camps.

Indicador de còpia: 1 bit. En cas de fragmentació, l'opció es copiarà o no a cada nou fragment segons el valor d'aquest camp:

0 = no es copia
1 = es copia.

Classe d'opció: 2 bits. Els possibles classes són:

0 = control
1 = reservada
2 = depuració i mesuraments
3 = ja aquesta.

Nombre d'opció: 5 bits. Identificador de l'opció.

Format de un paquet UDP.

Port d' origen	Port de destí	Nombre de seqüència	Nombre ACK	Longitud de la capçalera	Reservat	Control	Finestra	Checksum	Urgent Pointer	Opcions TCP
16 bits	16 bits	32 bits	32 bits	4 bits	4 bits	6 bits	16 bits	16 bits	16 bits	32 bits

Els **camps port d'origen i destinació** identifiquen l'aplicació en el terminal d'origen i destinació respectivament.
Nombre de seqüència identifica el primer byte del camp de dades, ja que en elTCP no es numeren segments, sinó bytes.
Nombre ACK menys un indica l'últim byte reconegut.
Longitud de capçalera indica això mateix, pot ser de 20 bytes, però si s'utilitza el camp opcions, pot arribar a 60 bytes.
Reservat com indica el seu nomo està reservat i se solen utilitzar zeros.
Control està format per 6 indicadors independents: URG (urgent), ACK (si es a 1 al camp número ACK té sentit), PSH (diu l' receptor si està a 1 que enviï totes les dades encara que la memòria intermèdia no estigui completament plena), RST (reset de la connexió), SYN (inici i re sincronització de la connexió) i FI (fi de la transmissió).
Finestra indica quants bytes componen la finestra de transmissió, en els protocols d'enllaç la mida de la finestra era constant, però ara un extrem TCP adverteix l'altre extrem de la quantitat de dades que està disposat a rebre en cada moment.
El camp **checksum**, com sempre detecta errors.
Urgent Pointer té sentit quan URG està a 1 i indica que les dades que envia l'origen són urgents i identifica l'últim byte del camp de dades que també ho és.
Opcions TCP permet afegir camps a la capçalera per marcar el temps en què es va transmetre el segment, augmentar la mida de la finestra, indicar la mida màxima del segment MSS.

Format de un paquet TCP.

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

Port origen (16 bits): Identifica el port emissor.
Port destinació (16 bits): Identifica el port receptor.
Aquests dos valors identifiquen l'aplicació receptora i l'emissora, juntament amb les adreces IP de l'emissor i receptor identifiquen de forma unívoca cada connexió. La combinació d'una adreça IP i un port és anomenat socket. És el parell de sockets (adreça IP + port de l'emissor i adreça IP + port del receptor) emissor i receptor el que especifica els dos punts finals que unívocament es corresponen amb cada connexió TCP a internet.
Nombre de seqüència (32 bits): Identifica el byte del flux de dades enviat per l'emissor TCP al receptor TCP que representa el primer byte de dades del segment.
Si considerem un flux de bytes unidireccional entre les dues aplicacions, TCP numera cada byte amb un nombre de seqüència. Aquest nombre de seqüència és de 32 bits sense signe que retorna a 0 en arribar a 232 -1. Quan una connexió està sent establerta el flag SYN s'activa i el camp del nombre de seqüència conté l'ISN (initial sequence number) triat pel host per a aquesta connexió. El nombre de seqüència del primer byte de dades serà l'ISN +1 ja que el flag SYN consumeix un nombre de seqüència.
Nombre de justificant de recepció (32 bits): Conté el valor del següent número de seqüència que l'emissor del segment espera rebre.
Una vegada que la connexió ha estat establerta, aquest nombre s'envia sempre i es valida amb el flag ACK activat. Enviar ACKs no costa res ja que el camp de justificant de recepció sempre forma part de la capçalera, igual que el flag ACK. TCP es pot descriure com un protocol sense assentaments selectius o negatius ja que el nombre d'assentiment en la capçalera TCP significa que s'han rebut correctament els bytes anteriors però no s'inclou aquest byte. No es poden assentir parts selectives del flux de dades (suposant que no estem usant l'opció SACK de assentiments

selectius). Per exemple si es reben correctament els bytes 1-1024 i el següent segment conté els bytes 2049-3072, el receptor no pot assentir aquest últim segment. Tot el que pot enviar és un ACK amb 1025 com a número d'assentiment, com si arriba el segment 1025-2048 però amb un error de cheksum.
Longitud de capçalera (4 bits): especifica la mida de la capçalera en paraules de 32 bits.
És requerit perquè la longitud del camp "opcions" és variable. Per tant la mida màxima de la capçalera està limitat a 60 bytes, mentre que sense "opcions" la mida normal serà de 20 bytes. A aquest camp també se li sol cridar "data offset" pel fet que és la diferència en bytes des del principi del segment fins al començament de les dades.
Reservat (3 bits): per a ús futur. Ha d'estar a 0.
Flags (9 bits):

NS (1 bit): ECN-nonce concealment protection. Per protegir davant paquets accidentals o maliciosos que s'aprofiten del control de congestió per guanyar ample de banda de la xarxa.
CWR (1bit): Congestion Window Reduced. El flag s'activa pel host emissor per indicar que ha rebut un segment TCP amb el flag ECE activat i ha respost amb el mecanisme de control de congestió.
ECE (1 bit): Per donar indicacions sobre congestió.
URG (1 bit): Indica que el camp del punter urgent és vàlid.
ACK (1 bit): Indica que el camp d'assentiment és vàlid. Els paquets enviats després del paquet SYN inicial han de tenir actiu aquest flag.
PSH (1 bit): Push. El receptor ha de passar les dades a l'aplicació tan aviat com sigui possible.
RST (1 bit): Reset. Reinicia la connexió.

Format de un paquet ICMP.

Tipus	Codi	Checksum	Dades
8 bits	8 bits	16 bits	

Tipus indica el caràcter del missatge enviat, ja que el protocol permet especificar una gran varietat d'errors o missatges de control de flux de les comunicacions.
Codi indica el codi d'error dins del tipus d'error indicat en el camp "tipus". És a dir, agrupa els missatges en tipus i dins de cada tipus especifica el codi concret a què es refereix.
Checksum simplement permet verificar la integritat del missatge enviat, el que permet detectar possibles errors en la tramesa, transport o recepció del missatge de control ICMP**.**